PENGELOMPOKAN EMOSI BERDASARKAN LIRIK LAGU MENGGUNAKAN METODE *ITERATIVE DICHOTOMIZER TREE*

SKRIPSI

Untuk memenuhi sebagian persyaratan
mencapai gelar Sarjana Komputer

Disusun oleh :

MOHAMMAD YASSER BURHAN

NIM. 0610963026

PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2013

BAB I
PENDAHULUAN

1.1 Latar Belakang

Lagu merupakan hiburan dan aktivitas manusia yang melibatkan suara-suara yang teratur. Karena lagu berupa sekumpulan nada-nada yang dirangkai menjadi sebuah bunyi yang sangat indah dan harmoni. Dalam penulisan sebuah lagu, terdapat istilah yang biasa digunakan dalam menulis lagu seperti, *interlude* yang merupakan bagian kosong pada lagu, dan *reff* yang berarti pengulangan.

Bagian dari lagu yang berperan dalam membangun emosi adalah lirik lagu. Rangkaian pemilihan kata-kata yang tepat diperlukan dalam membangun sebuah emosi yang terpancar dari sebuah lagu. Karena itu, klasifikasi lagu berdasarkan keterkaitan antara lagu dengan emosi telah banyak digunakan berbagai kesempatan oleh manusia dilakukan secara manual. Misalnya, pada film, lagu digunakan untuk mempertegas suasana pada *scene-scene* tertentu: lagu yang dramatis digunakan untuk melatarbelakangi *scene* yang menegangkan, lagu yang bersemangat untuk *scene* perang, lagu yang menyenangkan digunakan sebagai latar belakang *scene* humor [AKB-08].

Seiring banyaknya lagu maka semakin beragam emosi yang muncul, sehingga sulit mencari lagu yang sesuai dengan emosi yang diinginkan. Dan dengan banyaknya lirik lagu membutuhkan *cost* yang lebih tinggi dalam pengkategorian secara manual, maka diperlukan suatu pengkategorian otomatis untuk menyelesaikan masalah tersebut.

Teks mining adalah salah satu bidang khusus dari data mining. Teks mining merupakan sebuah teknologi baru yang dapat digunakan untuk menambang data yang telah ada dalam sebuah *database* dengan membuat suatu data berupa teks yang tidak terstruktur menjadi data yang dapat dianalisa [FRA-10].

Text mining sendiri dapat dibedakan menjadi dua menurut teknik pembelajaran yaitu, *unsupervised learning* dan *supervised learning*. Pada penelitian ini digunakan metode *supervised learning* dengan menggunakan algoritma *Iterative Dichotomizer Tree.* Pengklasifikasian dengan algoritma *Iterative Dichotomizer Tree* dengan membangkitkan pohon keputusan dari

sejumlah data *training*. Pada pengklasifikasian lirik lagu terdapat dua jenis data, data *training* dan data *testing*. Data *training* adalah data yang sudah memilih label katagori tertentu dan data *testing* adalah data yang digunakan dalam proses pengklasifikasian berdasarkan proses pelatihan.

Berbagai penelitian mengenai teks mining telah dilakukan dengan beberapa metode, antara lain dengan menggunakan algoritma *Iterative Dichotomizer Tree*, *K-Nearest Neighbor*, *NaÏve Bayes Classifier*, dan *Neural Network* [RAM-07]. Hasil eksperimen dari penelitian klasifikasi teks secara benar ke dalam 20 kategori *newsgroup*, *Iterative Dichotomizer tree* dengan nilai rata–rata akurasi 79,48 % dan waktu rata–rata 0,5 menit, *K-Nearest Neighbor* dengan nilai rata–rata akurasi 40,72 % dan waktu rata-rata 2 menit, *NaÏve Bayes Classifier* dengan nilai rata-rata akurasi 74,56 % dan waktu rata–rata 0,5 menit, dan *Neural Network* dengan nilai rata-rata akurasi 31,18 % dan waktu rata–rata 9 menit. Metode pembentukan pohon *Iterative Dichotomizer Tree* merupakan algoritma yang terbaik untuk melakukan klasifikasi dokumen teks, dengan nilai rata - rata tingkat akurasi yang didapatkan dari hasil eksperimen dari penelitian klasifikasi teks secara benar ke dalam 20 kategori *newsgroup* yang mendekati 80% [RAM-07]. Dengan menerapkan metode *Iterative Dichotomizer Tree* dalam klasifikasi emosi yang sama–sama menggunakan dalam bentuk *text*, diharapkan memiliki tingkat akurasi yang tinggi. Berdasarkan latar belakang, maka digunakanlah judul **"Pengelompokan Emosi Berdasarkan Lirik Lagu menggunakan Metode *Iterative Dichotomizer Tree*"**.

1.2 Rumusan Masalah

Rumusan masalah dalam skripsi ini adalah :

1. Bagaimana implementasi metode *Iterative Dichotomizer Tree* untuk menyelesaikan masalah pengklasifikasian emosi berdasarkan lirik lagu bahasa Inggris.
2. Bagaimana tingkat akurasi metode *Iterative Dichotomizer Tree* jika digunakan dalam pengklasifikasian lirik emosi berdasarkan lirik lagu bahasa Inggris.

1.3 Tujuan Penelitian

Beberapa tujuan yang ingin dicapai dalam penelitian dan penulisan skripsi ini yaitu :

1. Mengimplementasikan metode *Iterative Dichotomizer Tree* untuk menyelesaikan masalah pengklasifikasian emosi untuk lirik lagu berbahasa Inggris.
2. Menghitung tingkat akurasi yang diperoleh dari metode *Iterative Dichotomizer Tree* terhadap pengklasifikasian emosi untuk lirik lagu berbahasa Inggris.

1.4 Batasan Masalah

Batasan masalah yang diangkat dalam skripsi ini adalah :

1. Lirik yang digunakan memiliki format **.txt*.
2. Hanya menangani lirik lagu dalam bahasa Inggris.
3. Proses *stemming* yang digunakan adalah *porter stemmer*.
4. Terdiri dari 4 kategori emosi, yaitu *angry*, *love*, *fun*, *sad*.
5. Tingkat keberhasilan diukur dari besarnya nilai akurasi yang dihasilkan.
6. Proses penelitian dilakukan secara *offline*.
7. Frekuensi kemunculan kata tidak memperhatikan posisi kata itu berada pada bagian judul, *interlude* lirik, ataupun *reff* lirik.
8. Tidak membahas emosi berupa tanda baca dalam lirik.
9. Tidak memperhitungkan keterkaitan antar kata dalam lirik.

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat menghasilkan sistem pengkategorian otomatis untuk memudahkan pengguna dalam mencari lagu yang sesuai dengan emosi yang diinginkan sehingga *cost* untuk pengkategorian lebih rendah.

1.6 Metodologi Penelitian

Untuk mencapai tujuan yang dirumuskan sebelumnya, maka metodologi penelitian yang digunakan dalam penulisan skripsi ini adalah:

1. Studi Literatur

   Mempelajari dan mengkaji beberapa literatur (buku, jurnal ilmiah, dan artikel dari *website*) mengenai data mining, metode pengklasifikasian dokumen.

2. Perancangan dan implementasi sistem

   Membuat perancangan perangkat lunak dengan analisis terstruktur dan mengimplementasikan hasil rancangan tersebut.

3. Uji coba dan analisis hasil implementasi

   Akurasi hasil pengklasifikasian data dengan menggunakan model klasifikasi yang dibentuk dari data *training* serta akurasi kesalahan.

1.7 Sistematika Penulisan

Pembuatan skripsi ini dilakukan dengan pembagian bab sebagai berikut :

1. BAB I: PENDAHULUAN

   Berisi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

2. BAB II: KAJIAN PUSTAKA DAN DASAR TEORI

   Berisi kajian pustaka dan teori tentang *text mining*, metode ID3 dan teori-teori yang berhubungan dengan metode pengklasifikasian.

3. BAB III: METODE PENELITIAN DAN PERANCANGAN

   Berisi metode-metode yang digunakan dalam pembuatan sistem pengklasifikasian emosi berdasarkan lirik lagu kedalam suatu kategori dengan mengunakan pengklasifikasian ID3.

4. BAB IV: IMPLEMENTASI

   Berisi tentang penjelasan implementasi sistem.

5. BAB V: PENGUJIAN DAN ANALISIS

   Membahas proses dan melakukan analisa terhadap hasil pengujian.

6. BAB VI: PENUTUP

   Berisi kesimpulan yang diperoleh dari hasil pengujian dan saran-saran untuk pengembangan.

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

2.1 Kajian Pustaka

2.1.1 Jurnal Klasifikasi *E-mail Spam* Berbahasa Inggris

Jurnal Klasifikasi *E-mail Spam* Berbahasa Inggris dibuat oleh Nugraha. Welly. S dari Universitas Brawijaya, Malang. Jurnal tersebut melaporkan tentang penelitian klasifikasi *E-mail Spam* Berbahasa Inggris menggunakan algoritma yang diajukan yaitu *Iterative Dichotomizer Tree* dan dibandingkan dengan algoritma-algoritma yang sudah ada seperti *Naive Bayes, Nearest Neighbor*, dan *Neural Network.*

Sumber data latih pada penelitian tersebut diambil dari *e-mail spam* didapatkan dari *e-mail repository* dari *http://untroubled.org/spam/*, dan *e-mail* non-*spam* didapatkan dari *http://spamassassin.apache.org/publiccorpus/* dengan total dokumen sebanyak 100 yang termasuk dalam 2 kategori: *E-mail Spam* dan *E-mail non-Spam.*

Seleksi fitur pada penelitian tersebut menggunakan pendekatan *Decision Tree.* Pada pendekatan ini nilai bobot diubah kedalam bentuk data berkategori lalu dipilih *Information Gain* yang tertinggi.

Hasil keakuratan algoritma *Dichotomizer Tree* diperkirakan dapat mencapai 90%.

2.2 Emosi Lagu

2.2.1 Definisi Emosi Lagu

Lagu merupakan hiburan dan aktivitas manusia yang melibatkan suara-suara yang teratur. Karena lagu berupa sekumpulan nada-nada yang dirangkai menjadi sebuah bunyi yang sangat indah dan harmoni.

Emosi lagu menjelaskan makna emosional yang melekat pada sebuah klip lagu. Hal ini membantu dalam pemahaman lagu, pencarian lagu dan beberapa aplikasi yang berkaitan dengan lagu [LIU-03].

Klasifikasi lagu berdasarkan emosi dapat dilakukan secara manual dan subyektif oleh manusia, seperti yang banyak dilakukan pada kasus pemilihan lagu

untuk latar belakang suatu film. Biasanya pada tim pembuat film tersebut terdapat tim kecil yang khusus bertugas untuk menangani masalah pemilihan, bahkan pembuatan, lagu latar belakang. Dapat dilihat pada film-film yang telah dibuat bahwa emosi yang digerakkan oleh lagu latar tersebut seringkali benar-benar mengena sesuai dengan emosi yang diinginkan pada *scene-scene* tertentu pada film tersebut [AKB-08].

2.2.2 Jenis Lagu

Jenis lagu yang diambil dari situs *www.allmusic.com*, *www.popculturemadness.com*, dan *www.stereomood.com* dikelompokkan berdasarkan emosi lagu menggunakan *multi-label* klasifikasi. Lebih rincinya ditunjukkan pada tabel 2.1.

Tabel 2.1 Jenis Lagu

Nama Situs	Kategori	Jumlah data
www.allmusic.com	angry	50
	love	50
	fun	50
	sad	50
www.popculturemadness.com	angry	51
	love	75
	fun	125
	sad	100
www.stereomood.com	angry	1049
	love	2421
	fun	6111
	sad	3993

2.2.3 Bagian Lagu

Berikut adalah bagian-bagian yang sering terdapat dalam sebuah lagu:

1. Intro

Intro merupakan pengawalan lagu masuk, kebanyakan dari intro berupa instrumen yang not-notnya diambil dari bagian lagu tersebut. Kata lainnya intro adalah pembukaan sebelum mulai lagu.

2. Bait / *Verse*

Bait atau *verse* merupakan awalan dari sebuah lagu. Penulisannya terkadang memakai bait 1, bait 2, dan seterusnya, bait merupakan titik awal penceritaan lagu.

3. *Bridge*

Umumnya merupakan 'penyela' antara *verse* dan *chorus*.

4. *Chorus*

Umumnya dikenal sebagai *reff* (walaupun sebetulnya *reff* memiliki pengertian yang berbeda). *Chorus* memiliki nilai *excitement* yang lebih tinggi dari *verse*, dan sering diasosiasikan sebagai puncak dari sebuah lagu. Biasanya *statement* utama lagu ada di bagian ini. Melodi *chorus* biasanya sudah merupakan pengembangan lebih lanjut dari *verse*, yang mengandung lompatan klimaks.

Chorus merupakan bagian inti dari sebuah lagu biasanya merupakan puncak dari lagu, biasanya bagian ini yang merupakan isi lagu yang ditunggu-tunggu untuk didengarkan. Biasanya bagian *chorus* atau *reff* ini dilagukan berulang-ulang untuk memberi tahu pendengar inti lagu tersebut. Kebanyakan dari *reffrain* notasi pengulangannya sama dan syairnyapun sama, namun tidak menutup kemungkinan syairnya sedikit dimodifikasi, cuman biasanya tak jauh dari *reff* yang pertama, atau istilah lainnya beda-beda tipis.

5. *Reffrain / Reff*

Sebetulnya istilah *reff* masuk dalam konteks aransemen. Arti dari *reff* adalah *reffrain*, atau berarti 'Pengulangan'. Maksudnya ada bagian lagu yang dinyanyikan berulang-ulang. Dalam hal ini biasanya *chorus*-lah yang dinyanyikan berulang-ulang, makanya seringkali istilah *chorus* 'tertukar' dengan *reff*. Padahal tidak semua *reff* merupakan *chorus*.

6. Solo Instrument

Bagian ini merupakan bagian sang pemain *instrument* menunjukkan permainan *instrument* tanpa diselingi oleh suara penyanyi. Jika ada suara penyanyi hanya sebagai pengisi suara latar saja yang lebih mengedepankan permainan *instrument* lagu.

7. Interlude

Interlude itu bagian kosong pada lagu seperti layaknya '*intro'* yang berada di tengah–tengah lagu. *Interlude* ini bagian yang menyambungkan bait dengan bait atau bait dengan *chorus*. Tidak terdapat syair dalam *interlude* ini. *Interlude* hanya terdiri dari beberapa baris atau pola *chord*. Mungkin 4 baris, 6 baris atau 8 baris.

8. Ending

Ending merupakan bagian lagu yang paling akhir, mengacu pada lagu-lagu yang sudah ada biasanya berupa *fade out* atau *looping*, ataupun lagu akan berhenti dibaris terakhir.

2.3 Teks Mining

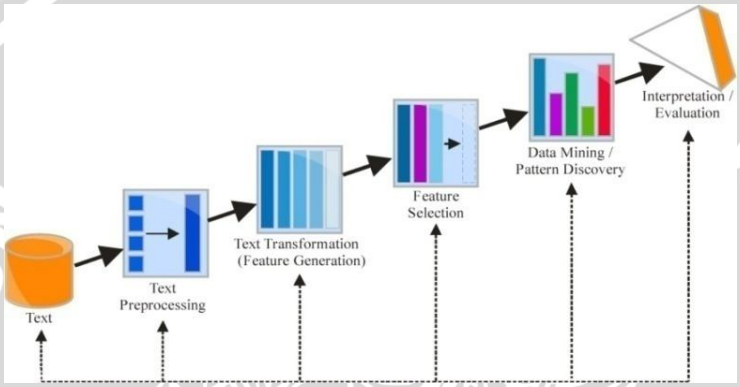
Teks mining adalah salah satu bidang khusus dari data mining. Teks mining merupakan sebuah teknologi baru yang dapat digunakan untuk menambang data yang telah ada dalam sebuah *database* dengan membuat suatu data berupa teks yang tidak terstruktur menjadi data yang dapat dianalisa [FRA-10].

Teks mining juga dapat didefinisikan sebagai suatu proses menggali informasi dimana seorang *user* berinteraksi dengan sekumpulan dokumen menggunakan *tools* analisis yang merupakan komponen-komponen dalam data mining yang salah satunya adalah kategorisasi. Tujuan dari *text mining* adalah untuk mendapatkan informasi yang berguna dari sekumpulan dokumen. Jadi, sumber data yang digunakan pada *text mining* adalah kumpulan teks yang memiliki format yang tidak terstruktur atau minimal semi terstruktur. Adapun tugas khusus dari *text mining* antara lain yaitu pengkategorisasian teks *(text categorization)* dan pengelompokan teks *(text clustering)* [MOO-06].

Salah satu elemen kunci dari *text mining* adalah kumpulan dokumen yang berbasis teks. Pada prakteknya, *text mining* ditujukan untuk menemukan pola dari sekumpulan dokumen yang jumlahnya sangat besar dan bisa mencapai jumlah ribuan bahkan sampai jutaan. Koleksi dokumen bisa statis, dimana dokumen tidak berubah, dimana dokumen selalu di-*update* sepanjang waktu [TRI-09].

2.4 Tahap Teks Mining

Dari definisi diatas, dalam melakukan teks mining terdapat tahapan-tahapan proses untuk mendapatkan pengetahuan, tahapan-tahapan itu ada lima yaitu: *pre-processing*, *teks transformation*, *feature selection*, data mining, dan *interpretation/evaluation*. Namun secara lengkap hanya tiga yang digunakan dalam teks mining yaitu *pre-processing*, *teks transformation*, dan *pattern discovery* atau data mining [EVE-02]. Penggambaran tahap teks mining menurut Evan, Yahir dan Zohar dapat dilihat pada gambar 2.1.



Gambar 2.1 Tahap dalam teks mining [EVE-02]

2.4.1 Tahap *Preprocessing*

Pada tahap *preprocessing* dilakukan proses pengolahan teks menjadi data yang akan digunakan dalam proses selanjutnya. Tujuan dilakukannya *preprocessing* untuk melakukan seleksi informasi yang tidak diperlukan dalam proses selanjutnya seperti *tag-tag HTML*, kata umum yang biasanya muncul dalam jumlah besar tetapi tidak memiliki makna dan pembuangan imbuhan kata untuk diambil kata dasar dari setiap kata yang berimbuhan. *Preprocessing* penting dilakukan untuk meningkatkan keakurasian dalam proses klasifikasi. Pada preproses terdapat proses pengolahan teks seperti: *Tokenizing*, *stemming*, penghapusan *stopwords* dan frekuensi kata.

2.4.1.1 *Preprocessing* Tahap *Tokenizing*

Tahap *tokenizing* adalah tahap pemotongan *string input* berdasarkan tiap kata yang menyusunnya [MOO-06]. Pada tahap *tokenizing* dilakukan dengan

menghilangkan tanda baca dan memisahkan per spasi. Dengan *tokenizing*, dapat mempermudah dan mempercepat pra proses teks mining.

2.4.1.2 *Preprocessing* Tahap *Stemming*

Tahap *stemming* adalah tahap mencari *root* kata dari tiap kata hasil *filtering* [MOO-06]. Pengertian *stemming* bisa juga diartikan sebagai proses mengembalikan kata menjadi kata dasarnya [BUD-08]. Proses pencarian kata dasar dapat dilakukan dengan membuang awalan kata dan akhiran kata yang terdapat dalam kata berimbuhan. Implementasi proses *stemming* beragam tergantung dengan bahasa dari dokumen. Dengan dilakukanya proses *stemming* setiap kata berimbuhan akan berubah menjadi kata dasar, dengan demikian dapat lebih mengoptimalkan proses *teks mining*.

2.4.1.3 *Preprocessing* Tahap *Stopword*

Stopword adalah kata umum (*common words*) yang biasanya muncul dalam jumlah besar dan dianggap tidak memiliki makna [WIB-08]. Tahap *stopword* dilakukan dengan pembuangan kata-kata yang sering muncul seperti kata sambung, kata tanya dan kata-kata lain yang tidak memiliki makna. *Stopword* yang digunakan harus sesuai dengan dokumen yang akan diproses.

Dalam *teks mining* proses *stemming* dan *stopword* dapat dilakukan dalam proses berbeda atau dalam satu proses tergabung. Dilakukan proses penghilangan *stop word* terlebih dahulu kemudian dilanjutkan proses *stemming*.

2.4.1.4 *Preprocessing* Tahap Frekuensi Kata

Proses terakhir dari *preprocessing* adalah frekuensi kata. Dalam frekuensi kata dilakukan proses perhitungan kemunculan kata pada setiap dokumen. Kata-kata yang memiliki frekuensi tertentu menjadi ciri dari setiap dokumen. Oleh karena itu, hanya kata-kata yang muncul minimal 3 kali didalam keseluruhan dokumen akan digunakan sebagai ciri dari masing-masing dokumen [JOA-98].

2.4.2 Tahap Teks *Transformation* Dan *Feature Generation*

Feature adalah suatu pola menarik dari dokumen yang dianggap bisa merepresentasikan suatu dokumen. Tidak semua *feature* bermanfaat dalam proses selanjutnya. Tujuan dilakukannya proses ini untuk mengambil *feature-feature* yang bermanfaat. Tahap *feature generation* dapat dilakukan dengan bermacam-macam metode, bergantung pada kebutuhan. Pada skripsi ini tahap *feature generation* dengan mengunakan pembobotan *TF-IDF (term frequency-inverse document frequency*). Frekuensi dari sebuah *term* dalam satu dokumen dapat merepresentasikan makna sebuah *term* pada suatu dokumen.

2.4.3 Tahap Data mining Atau *Pattern Discovery*

Tahap penemuan pola atau *pattern discovery* adalah tahap terpenting dari seluruh proses teks mining, tahap ini berusaha menemukan pola atau pengetahuan dari keseluruhan teks [MUS-09]. *Feature* yang lolos pada tahap sebelumnya akan memasuki tahap data mining untuk menghasilkan pola menarik dari data.

Pada tahap *pattern discovery* terdapat dua teknik pembelajaran, *unsupervised* dan *supervised learning*. Perbedaan dari kedua pembelajaran tersebut, *supervised learning* terdapat tabel atau nama kelas pada data latih dan data diklasifikasikan berdasarkan data latih.

Unsupervised learning tidak terdapat tabel atau nama kelas pada data latih, data dikelompokkan berdasarkan ukuran kemiripan pada suatu kelas.

Berdasarkan keluaran dari fungsi, *supervised learning* dibagi menjadi 2, regresi dan klasifikasi. Regresi terjadi jika *output* dari fungsi merupakan nilai yang kontinyu sedangkan klasifikasi terjadi jika keluaran dari fungsi adalah nilai tertentu dari suatu atribut tujuan (tidak kontinyu). Tujuan dari *supervised learning* adalah untuk memprediksi nilai dari fungsi untuk sebuah data masukan yang sah setelah melihat sejumlah data latih [LUZ-06].

Berikut adalah tahapan umum yang biasanya dilakukan pada *supervised learning* :

1. Menentukan tipe dari data latih.

2. Mengumpulkan data latih. Data latih yang digunakan seharusnya memiliki karakterisktik dunia nyata. Karena itu data latih dapat berasal baik dari hasil pengukuran atau dari pakar.
3. Menentukan representasi fitur masukan dari fungsi yang ingin dibentuk karena tingkat akurasi dari fungsi dapat dipengaruhi oleh representasi dari masukan.
4. Menentukan struktur dari pengetahuan (fungsi) dan algoritma yang akan digunakan.
5. Jalankan algoritma terhadap data latih.

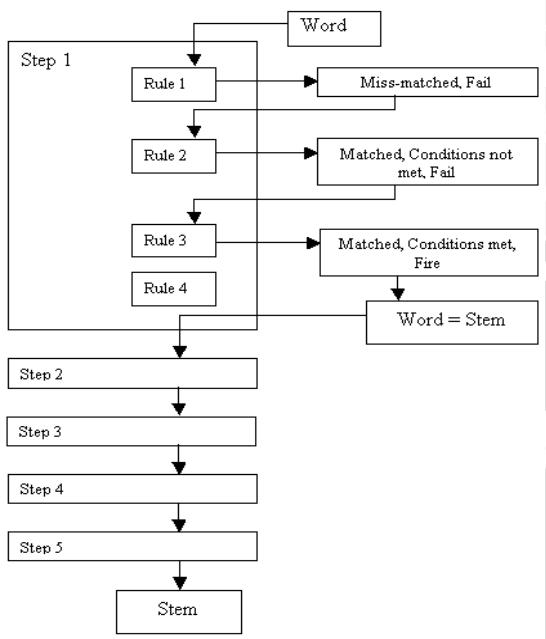
Terdapat banyak teknik *supervised learning* yang telah dikembangkan oleh para ahli namun sesuai dengan cakupan dari Skripsi ini, maka pada bahasan selanjutnya hanya akan dibahas *Iterative Dichotomizer Tree*.

2.5 *Stemming*

Tahap *stemming* adalah tahap mencari *root* kata dari tiap kata hasil *filtering* [HAR-06]. *Stem* (akar kata) adalah bagian dari kata yang tersisa setelah dihilangkan imbuhannya (awalan dan akhiran). Contoh : *connect* adalah *stem* dari *connected*, *connecting*, *connection*, dan *connections*. [ELD-08]. Pada tahap ini dilakukan proses pengembalian berbagai bentukan kata ke dalam suatu representasi yang sama. Tahap ini kebanyakan dipakai untuk teks berbahasa inggris dan lebih sulit diterapkan pada teks berbahasa Indonesia. Hal ini dikarenakan bahasa Indonesia tidak memiliki rumus bentuk baku yang permanen [TRI-09].

2.5.1 Algoritma *Porter Stemmer*

Algoritma *Porter Stemmer* pertama kali ditampilkan pada tahun 1980 dan dikembangkan oleh Martin Porter di Universitas Cambridge. *Porter Stemmer* adalah sufiks konteks algoritma penghapusan sensitif yang paling banyak digunakan dari semua *stemmer* dan implementasi dalam berbagai bahasa yang tersedia [HOO-05]. Algoritma *Porter Stemmer* adalah proses untuk menghapus bentuk biasa dan akhiran yang berinfleksi dari sebuah kata bahasa Inggris. Kegunaan utamanya adalah sebagai bagian dari aturan proses normalisasi yang biasanya dilakukan ketika mengatur *Information Retrieval Systems* [POR-80].



Gambar 2.2 *Porter Stemmer* [HOO-05]

Algoritma *Porter Stemmer* didasarkan pada gagasan bahwa akhiran dalam bahasa Inggris (sekitar 1200) sebagian besar terdiri dari kombinasi sufiks yang lebih kecil dan sederhana. *Porter Stemmer* merupakan tahap *linear stemmer*. Secara khusus, *stemmer* tersebut mempunyai lima langkah untuk menerapkan aturan dalam setiap langkahnya seperti digambarkan pada gambar 2.2. Dalam setiap langkah, jika aturan akhiran dicocokkan dengan sebuah kata, maka kondisi yang melekat pada aturan tersebut akan diuji menjadi *stem* hasil, jika akhiran itu telah dihapus dengan cara yang ditentukan oleh aturan. Misalnya seperti kondisi yang mungkin, jumlah karakter vokal, yang diikuti menjadi karakter konsonan dalam *stem* (ukuran), harus lebih besar dari satu untuk aturan yang akan diterapkan.

Setelah melewati kondisi aturan dan diterima, aturan dibebaskan dan akhiran akan dihapus dan berpindah ke langkah berikutnya. Jika aturan tidak diterima maka aturan selanjutnya akan diuji, sampai aturan dari langkah tersebut dilepaskan dan kontrol lolos ke tahap berikutnya atau tidak ada aturan dalam langkah tersebut yang mengontrol perpindahan ke langkah berikutnya. Proses ini

berlanjut untuk kelima langkah, resultan *stem* yang dikembalikan oleh *stemmer* setelah kontrol telah berlalu dari langkah kelima.

Algoritma *Porter Stemmer* merupakan *stemmer* yang sering dipakai dan digunakan dalam berbagai aplikasi. Implementasi *stemmer* tersedia di *situs* yang didirikan oleh Porter sendiri, dengan implementasi menggunakan *Java, C* dan *PERL*, *website* tersebut juga termasuk salinan dari *paper* yang mendefinisikan algoritma tersebut. Implementasi lain dari algoritma ini tersedia dalam *web* tersebut. Algoritma *Porter stemmer* mungkin adalah yang paling banyak digunakan dalam penelitian mengenai *IR* [HOO-05].

2.5.2 Algoritma *Porter Stemmer* untuk Lirik Bahasa Inggris

Porter Stemmer merupakan algoritma penghilangan akhiran *morphological* dan infleksional yang umum dari bahasa Inggris. Aturan dalam algoritma *Porter* dipisahkan menjadi 5 tahap yang berbeda yang diberi nomor dari 1 sampai 5. Tahap-tahap tersebut diterapkan pada kata-kata dalam teks mulai dari tahap 1 sampai ke tahap 5. Selanjutnya, tahapan tersebut akan diterapkan secara berurutan satu persatu setelah tahap yang lain sebagai perintah dalam program.

Sebelum dilakukan kelima tahapan, terlebih dahulu ditentukan kondisi-kondisi atau *action rules*-nya. Kondisi dikelompokkan menjadi 3 kelas, yaitu:

- Kondisi pada *stem*

Ukuran (*measure*), dinotasikan dengan *m*, dari sebuah *stem* berdasarkan pada urutan vokal-konsonan.

Sebelum melakukan *stemming* langkah pertama yang dilakukan adalah menentukan *measure*. Cara menentukan *measure* adalahmenentukan *consonant* dan *vowel* terlebih dahulu. *Consonant* merupakan huruf mati seperti “D” , “Z”, “H”, sedangkan *vowel* merupakan huruf vokal seperti “A”,”E”,”I”.

Setiap barisan *consonant* akan dinotasikan menjadi “C” sedangkan barisan *vowel* dinotasikan menjadi “V”. Contohnya adalah kata berikut: “FORK” , kata tersebut memiliki bentuk CVC, C pertama adalah “F” , V adalah “O”, dan C kedua adalah “RK”.

Karena setiap kata dalam bahasa Inggris merupakan kumpulan dari bentuk CVCVCV.... atau VCVCVC.... untuk mempermudah dibentuklah notasi $[C](VC)^m[V]$. Bagian $(VC)^m$ menandakan bahwa VC diulang sebanyak *m* kali, *m* inilah yang nantinya disebut dengan *measure.* Berikut ini adalah contoh-contoh *measure*:

Tabel 2.2 Kondisi pada *stem*

Kondisi	Contoh
m=0	TR, EE, TREE, Y, BY
m=1	TROUBLE, OATS, TREES, IVY
m=2	TROUBLES, PRIVATE, OATEN

- X> berarti *stem* berakhir dengan huruf X
- *v* berarti *stem* mengandung sebuah vokal
- *d berarti *stem* diakhiri dengan konsonan *doubel*
- *o berarti *stem* diakhiri dengan konsonan-vokal-konsonan, berurutan, di mana konsonan akhir bukan w, x, atau y.

- Kondisi pada *suffix*

  (current_suffix == pattern)

- Kondisi pada *rule*

  Rule-rule dibagi menjadi beberapa *step*. *Rule-rule* dalam sebuah *step* diuji secara berurutan, dan hanya 1 *rule* dari suatu *step* yang diterapkan.

  Step (langkah-langkah) pada algoritma *Porter Stemmer* (*http://people.ischool.berkeley.edu/~hearst/irbook/porter.html*):

- *Step* 1a: ***remove plural suffixation***, yaitu menghapus/ mengganti akhiran pada kata yang berbentuk jamak, berupa akhiran *sses* menjadi *ss, ies* menjadi *i, ss* (tidak diganti), dan akhiran *s* hingga didapatkan *stem. Step 1a* dijelaskan pada tabel 2.3.
- *Step* 1b: ***remove verbal inflection,*** yaitu menghapus/ mengganti akhiran pada kata yang mengalami modulasi lisan/ pengucapan, berupa akhiran *eed* (jika terdapat paling kurang sebuah huruf vokal-konsonan berurutan) menjadi *ee*, serta akhiran *ed* dan *ing* (tidak diganti) untuk kata yang hanya memiliki sebuah huruf vokal, dan dihapus untuk yang memiliki lebih dari satu huruf vokal. *Step 1b* dijelaskan pada tabel 2.4.

- *Step* 1b1: ***continued for -ed and -ing rules***, berupa tahap lanjutan untuk *rule* akhiran *ed* dan *ing*. Hasil *stemming* pada akhiran *ed* dan *ing* pada *step* sebelumnya akan di-*stemming* lagi yaitu dengan menghapus kata yang berakhiran *at* (diganti menjadi *ate*), *bl* menjadi *ble*, *iz* menjadi *ize*, untuk kata yang diakhiri dengan *double* huruf konsonan dan tidak berakhir dengan huruf *l*, *s*, atau *z* akan diganti menjadi kata yang berakhir satu huruf konsonan saja, jika kata berakhir dengan huruf *l, s,* atau *z* maka tidak diganti, dan untuk kata yang diakhiri dengan huruf konsonan–vokal–konsonan berurutan, di mana konsonan akhir bukan *w, x,* atau *y* dan hanya terdapat satu urutan huruf vokal-konsonan di dalamnya maka ditambahkan *e*. *Step* 1b1 dijelaskan pada tabel 2.5.
- *Step* 1c: ***y and i,*** jika kata mengandung sebuah huruf vokal dan berakhiran y akan diganti dengan i. *Step* 1c dijelaskan pada tabel 2.6.
- *Step* 2: ***peel one suffix off for multiple suffixes***, dengan kata tersebut memiliki paling kurang sebuah huruf vokal-konsonan berurutan, yaitu kata berakhiran *ational* atau *ation* atau *ator* (menjadi *ate*), *tional* (menjadi *tion*), *enci* (menjadi *ence*), *anci* (menjadi *ance*), *izer* atau *ization* (menjadi *ize*), *iviti* atau *iveness (*menjadi *ive), ality* atau *alism* atau *alli* (menjadi *al*), *biliti* (menjadi *ble*), *abli* (menjadi *able*), *ently* (menjadi *ent*), *eli* (menjadi *e*), *ousli* dan *ousness* (menjadi *ous*), *fulness* (menjadi *ful*). *Step* 2 dijelaskan pada tabel 2.7.
- *Step* 3: dengan kata tersebut memiliki paling kurang sebuah huruf vokal-konsonan berurutan, kata berakhiran *ative* atau *ful* atau *ness* akan dihapus, kata berakhiran *icate* atau *iciti* atau *ical* (menjadi *ic*), *alize* (menjadi *al*). S*tep* 3 dijelaskan pada tabel 2.8.
- *Step* 4: ***delete last suffix,*** dengan kata tersebut memiliki paling kurang dua huruf vokal-konsonan berurutan, kata berakhiran *al, ance, ence, er, ic, able, ible, ant, ement, ment, ent, ion, ou, ism, ate, iti, ous, ive*, dan *ize* akan dihapus. *Step* 4 dijelaskan pada tabel 2.9.
- *Step* 5a: ***remove e ,*** menghapus akhiran *e* jika kata tersebut paling kurang memiliki dua huruf vocal-konsonan yang berurutan atau memiliki sebuah huruf vocal-konsonan berurutan dan tidak diakhiri dengan huruf konsonan–vokal–konsonan berurutan, di mana konsonan akhir bukan *w, x,* atau *y*. S*tep* 5a dijelaskan pada tabel 2.10.

- *Step* 5b: ***reduction ,*** jika kata hanya memiliki sebuah huruf vokal-konsonan berurutan dan tidak berakhir dengan dobel huruf konsonan dan huruf *l* maka diganti dengan akhiran satu huruf konsonan saja. *Step* 5b dijelaskan pada tabel 2.11.

Tabel 2.3 *Step* 1a: Menghapus akhiran jamak

Conditions	*Suffix*	*Replacement*	*Examples*
NULL	Sses	Ss	caresses → caress
NULL	ies , ied	I	ponies → poni Ties → tie
NULL	ss, us	Ss	caress → caress chorus → chorus
NULL	S	NULL	cats → cat

Tabel 2.4 *Step* 1b: Menghapus kata kerja verbal

Conditions	*Suffix*	*Replacement*	*Examples*
(m>0)	eed, eedly	Ee	Feed → feed Agreed → agree
(*v*)	ed, edly	NULL	Plastered → plastered Bled → bled
(*v*)	ing, ingly	NULL	Motoring → motor Sing → sing

Tabel 2.5 *Step* 1b1: Lanjutan untuk aturan *–ed* dan *–ing*

Conditions	*Suffix*	*Replacement*	*Examples*
NULL	At	Ate	confla(ed) → conflate
NULL	Bl	Ble	troubl(ing) → trouble
NULL	iz	Ize	siz(ed) → size
(*d and not (*<L> or *<S> or *<Z>))	NULL	Single letter	hopp(ing) → hop tann(ed) → tan fall(ing) → fall hiss(ing) → hiss fizz(ed) → fizz
(m=1 and *o)	NULL	E	fail(ing) → fail fil(ing) → file

Tabel 2.6 *Step* 1c: y dan i

Conditions	*Suffix*	*Replacement*	*Examples*
(*v*)	y	I	happy → happi sky → sky

Tabel 2.7 *Step* 2: Memisahkan satu akhiran untuk beberapa akhiran

Conditions	*Suffix*	*Replacement*	*Examples*
(m>0)	ational, ation, ator	Ate	Relational → relate Predication → predicate Operator → operate
(m>0)	tional	Tion	Conditional → condition Rational → rational
(m>0)	enci	Ence	Valenci → valece
(m>0)	anci	Ance	Hesitanci → hesitance
(m>0)	izer, ization	Ize	Digitizer → digitize Vietnamization → vietnamize
(m>0)	abli	Able	Conformabli → conformable
(m>0)	ently	Ent	Differently → different
(m>0)	eli	E	Vileli → vile
(m>0)	ousli, ousness	Ous	Analogousli → analogous Callousness → callous
(m>0)	alism, aliti, alli	Al	Feudalism → feudal Formaliti → formal Radicalli → radical
(m>0)	iveness, iviti	Ive	Devcisiveness → decisive Sensitiviti → sensitive
(m>0)	fulness, fulli	Ful	Hopefulness → hopefull
(m>0)	biliti, bli	Ble	Sensibiliti → sensible

Tabel 2.8 *Step* 3

Conditions	*Suffix*	*Replacement*	*Examples*
(m>0)	icate	Ic	Triplicate → triplic
(m>0)	ative	NULL	Formative → form
(m>0)	alize	Al	Formalize → formal
(m>0)	iciti	Ic	Electriciti → electric
(m>0)	ical	Ic	Electrical → electric
(m>0)	Ful	NULL	Hopeful → hope
(m>0)	ness	NULL	Goodness → good

Tabel 2.9 *Step* 4: Menghapus akhiran terakhir

Conditions	*Suffix*	*Replacement*	*Examples*
(m>1)	al	NULL	Revival → reviv
(m>1)	ance	NULL	Allowance → allow
(m>1)	ence	NULL	Inference → infer
(m>1)	er	NULL	Airliner → airlin
(m>1)	ic	NULL	Gyroscopic → gyroscop
(m>1)	able	NULL	Adjustable → adjust
(m>1)	ible	NULL	Defensible → defens
(m>1)	ant	NULL	Irritant → irrit
(m>1)	ement	NULL	Replacement → replac
(m>1)	ment	NULL	Adjustment → adjust
(m>1)	ent	NULL	Dependent → depend
(m>1)	ion	NULL	Adoption → adopt
(m>1)	ou	NULL	Homologou → homolog
(m>1)	ism	NULL	Communism → commun
(m>1)	ate	NULL	Activate → active
(m>1)	iti	NULL	Angulariti → angular
(m>1)	ous	NULL	Homologous → homolog
(m>1)	ive	NULL	Effective → effect
(m>1)	ize	NULL	Bowdlerize → bowdler

Setelah semua suffix dihilangkan kini menuju ke langkah terakhir yaitu *step* 5 yang diterangkan di tabel 2.10 dan tabel 2.11.

Tabel 2.10 *Step* 5a: Hapus *e*

Conditions	*Suffix*	*Replacement*	*Examples*
(m>1)	e	NULL	Probate → probat Rate → rate
(m=1 and not *o)	e	NULL	Cease → ceas

Tabel 2.11 *Step* 5b: Reduksi

Conditions	*Suffix*	*Replacement*	*Examples*
(m=1 and *d and *<L>)	NULL	Single letter	Controll → control Roll → roll

2.6 *Term Frequency-Inverse Document Frequency*

Setelah melalui *preprocessing text* dihasilkan berbentuk *token* yang terpisah dari kata yang lain dan sudah dalam bentuk dasar. Pada langkah selanjutnya kata-kata atau *term* akan dirubah kedalam bentuk numerik untuk diketahui bobot setiap kata dari satu dokumen ke dokumen lainya. *Term Frequency* sering di sebut *TF*, merupakan banyak kata yang keluar dari satu dokumen. *Invers Dokumen Frequency* yang sering dikenal dengan *IDF*, merupakan pembobotan kemunculan jumlah kata pada suatu dokumen. Metode *TF-IDF* merupakan metode

pembobotan dalam bentuk sebuah metode yang merupakan *integrasi* antar *term frequency (tf),* dan *inverse document frequency (idf)* [YAN-99]. Berikut persamaan untuk mencari bobot kata dengan metode *TF-IDF*.

$$w_{ij} = tf_{ij} \cdot idf \qquad (2.1)$$

$$idf = \log \frac{N}{df_j} \qquad (2.2)$$

Keterangan :

- W_{ij} = bobot kata *i* pada dokumen *j*
- N = jumlah koleksi dokumen
- tf_{ij} = jumlah kehadiran kata *i* yang akan dihitung bobotnya dalam dokumen *j*
- df_j = dokumen *j* yang mengandung kata yang akan dihitung bobotnya
- *idf = inverse document frecuency* didapat dari hasil *log.N/df*

2.7 *Transformation Data* dengan Distribusi Frekuensi

Dalam persoalan generalisasi, data numerik yang dihasilkan tidak dapat begitu saja dikelompokan dalam kategori tertentu. Metode distribusi frekuensi dapat digunakan untuk membantu dalam pembuatan kelas dan interval suatu kategori. Data yang didapat dari dataset bertipe numerik, sedangkan pengujian ini memerlukan data tipe kategori. Teknik yang digunakan untuk mengubah data numerik menjadi data kategori adalah teknik distribusi frekuensi [DEF-10].

Berikut beberapa langkah dalam menentukan interval kelas pada metode distribusi frekuensi [SUP-00].

1. Urutkan data, untuk mencari nilai terbesar dan terkecil dari data. Gunakan persamaan 2.3 untuk mencari *range*.

$$R = X_{\max} - X_{\min} \qquad (2.3)$$

R = Rentang antara data terbesar dengan data terkecil.

X_{max} = Nilai data terbesar dari kumpulan data.

X_{min} = Nilai data terkecil dari kumpulan data.

2. Menentukan banyak kelas yang akan digunakan dengan menggunakan persamaan 2.4.

$$K = 1 + 3{,}3 \log N \qquad (2.4)$$

K = Banyak kelas

N = Jumlah data observasi

3. Menentukan panjang interval setiap kelas, dengan menggunakan persamaan 2.5.

$$Interval = \frac{R}{K} \qquad (2.5)$$

Interval = Panjang *range* data dalam tiap kelas

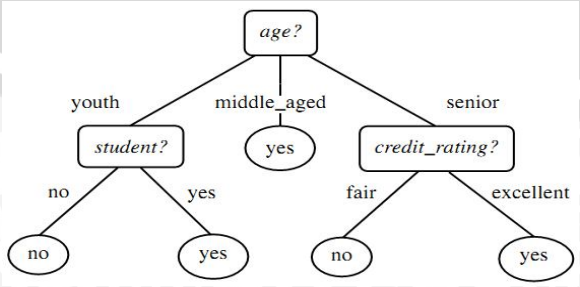
R = Rentang antara data terbesar dan terkecil

K = Banyak kelas

4. Pilih ujung bawah kelas interval pertama, dapat menggunakan data terkecil atau nilai data yang lebih kecil dari data.
5. Membuat daftar distribusi untuk mengetahui frekuensi masing-masing kelas.

2.8 *Decision Tree*

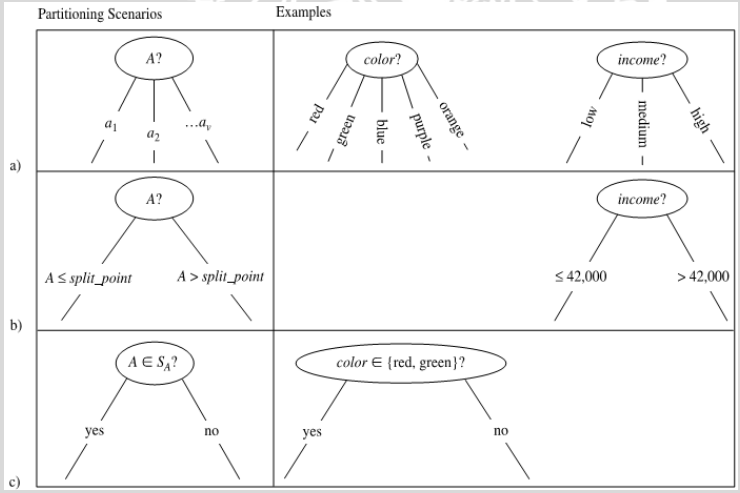
Penggunaan pohon keputusan dimulai pada tahun 1970-an dan awal 1980-an, J. Ross Quinlan, melakukan penelitian dan pengembangan algoritma ID3 (*Iterative Dichotomizer Tree*). Algoritma ini diperluas untuk konsep sistem pembelajaran yang dijelaskan oleh EB Hunt, J. Marin, dan P.T. Stone. Quinlan kemudian mempresentasikan C4.5 yang menjadi penerus ID3. Sekelompok ahli statistik mempublikasikan buku pengklasifikasian dan regresi *tree* (CART), yang menggambarkan pohon keputusan biner [BRE-84]. ID3 dan CART dikembangkan secara bebas pada waktu yang sama, namun mengikuti pendekatan yang sama untuk pembelajaran *decision tree* dari pelatihan *tuple*.



Gambar 2.3 *Decision tree* untuk pembelian komputer [HAN-06]

Decision tree adalah pembelajaran pohon keputusan dari *class* yang berlebel. *Decision tree* adalah struktur pohon seperti *flow chart*, dimana setiap *node internal (node non leaf*) menunjukkan tes pada atribut, setiap cabang berasal dari hasil pengujian, dan masing - masing daun (*node terminal*) memiliki label kelas. *Node* paling atas adalah *root.* Pada contoh *decision tree* digambar 2.3, pohon keputusan untuk pembelian komputer pada *AllElectronics*. Pada gambar tersebut menjelaskan apakah pelanggan *AllElectronics* memungkinkan untuk membeli komputer di tokoh tersebut. Setiap *node* (*non leaf*) internal merupakan tes dari atribut. Setiap simpul daun mewakili sebuah kelas, baik kelas berlabel *Yes* atau *No*.

Secara singkat bahwa *Decision Tree* merupakan salah satu metode klasifikasi pada *Text Mining*. Klasifikasi adalah proses menemukan kumpulan pola atau fungsi-fungsi yang mendeskripsikan dan memisahkan kelas data satu dengan lainnya, untuk dapat digunakan memprediksi data yang belum memiliki kelas data tertentu [HAN-06].



Gambar 2.4 *Decision tree* yang memungkinkan dalam proses partisi data [HAN-06]

Tiga kemungkinan untuk partisi data dalam *decision tree* berdasarkan kriteria pemisahnya. Ditunjukkan pada gambar 2.4, dengan contoh A sebagai atribut pemisah.

a) Jika A bernilai diskrit, salah satu cabang tumbuh untuk setiap nilai A. Penggunaan partisi seperti ini digunakan dalam pemodelan ID3, dijelaskan lebih lanjut pada sub bab berikutnya.
b) Jika A adalah *continuous_valued*, lalu dua cabang berkembang, sesuai dengan A $\leq$ *splite_point* dan A > *split_poin*. Penggunaan partisi dengan data split digunakan dalam pemodelan C4.5 dengan menggunakan *gain ratio*.
c) Jika A adalah nilai diskrit dan pohon biner harus diproduksi, maka tes bentuk A ϵ S_A, dimana S_A adalah pemisah subset untuk A. Penggunaan partisi seperti ini digunakan dalam pemodelan CART.

2.9 *Iterative Dichotomizer Tree*

Iterative Dichotomizer 3 (ID3) adalah algoritma *decision tree learning* (algoritma pembelajaran pohon keputusan) yang paling dasar. Algoritma ini melakukan pencarian secara rakus / menyeluruh (*greedy*) pada semua kemungkinan pohon keputusan [WAH-09].

ID3 menggunakan *informasi gain* sebagai ukuran seleksi atribut. Ukuran ini didasarkan pada penyelesaian *pionering* oleh *Claude Shannon* pada teori informasi, yang mempelajari nilai atau "informasi isi" dari pesan. Membiarkan *node* N mewakili atau terus mempartisi tupel D. Atribut dengan informasi *gain* tertinggi dipilih sebagai informasi pemisah *node* N. Atribut ini meminimalkan informasi yang dibutuhkan untuk mengklasifikasikan tupel dalam menghasilkan partisi dan menggambarkan keacakan atau ketidakmurnian dalam partisi ini. Informasi yang diharapkan untuk mengklasifikasikan sebuah *tuple* dalam D diberikan oleh [JIA-06].

Berikut persamaan untuk mencari perbandingan *entropy* atribut kategori dengan atribut kata.

$$Info\ (D) = -\sum_{i=1}^{m} p_i \log_2(p_i) \quad (2.6)$$

Keterangan :

- Info *(D)* = *Entropy* atribut untuk membandingkan data *sample*.
- *P* = Data *sample* ke *i* pada suatu atribut.

Dari hasil representasi *entropy* atribut yang ditunjuk sebagai *class*, dilakukan perhitungan seluruh *entropy* atribut untuk setiap data sampel. Data *sample* akan dibandingkan dengan data sampel dari atribut *class*. Dari hasil peritungan *entropy*, dilakukan perbandingan dengan *entropy class* dengan persamaan *information gain* sebagai berikut :

$$Gain\ (A) = Info\ (D) - Info_A\ (D) \quad (2.7)$$

Keterangan :

- *Gain (A)* = *Information gain* dari perbandingan *entropy class* dengan *entropy* setiap sampel atribut.
- *Info(D)* = perhitungan *entropy class.*
- $Info_A(D)$ = perhitungan *entropy* dari atribut.

Hasil dari *information gain* pada satu atribut dibandingkan dengan atribut lainnya, nilai informasi *gain* terbesar akan dijadikan *root.* Data sampel dari *root* utama menjadi cabang dari *tree*. Dilakukan rekursi untuk setiap cabang, dan memilih atribut terbaik dan dijadikan sebagai *node* cabang tersebut. Dilakukan rekursi sampai pada atribut terakhir untuk setiap cabang.

2.10 Evaluasi

Tujuan evaluasi percobaan pada *classifier* yaitu untuk mengukur keefektifannya apakah sistem mengklasifikasi secara benar. Evaluasi ini biasanya membutuhkan sebuah matrik yang disebut dengan Matriks *Confusion*. Matrik *Confusion* adalah sebuah matriks yang berisi tentang informasi mengenai hasil klasifikasi oleh sistem pengklasifikasi dan klasifikasi yang sebenarnya. Evaluasi standar yang biasa dilakukan adalah *precision* dan *recall*, sedangkan kombinasi dari kedua evaluasi tersebut adalah *F-measure*.

Tabel 2.12 Matrik *Confusion*

		Sistem	
		Positive	*negative*
Manual	*Positive*	*A*	*B*
	Negative	*C*	*D*

Dari tabel 2.12 dijelaskan kriteria pengkategorian data untuk penelitian, berikut penjelasan data pada tabel misalkan diberi katagori dengan C_m.

1. a = menyatakan jumlah dokumen yang dikategorikan sebagai kategori C_m oleh sistem yang juga merupakan hasil pengklasifikasian manual dengan kategori C_m.
2. b = menyatakan jumlah dokumen yang tidak dikategorikan benar/tidak relevan oleh sistem sebagai kategori C_m, namun sebenarnya pada pengklasifikasian manual berada pada kategori C_m.
3. c = menyatakan jumlah dokumen yang dikategorikan benar/relevan oleh sistem sebagai kategori C_m, namun sebenarnya bukan dari kategori C_m dari pengklasifikasian manual.

   d = menyatakan jumlah dokumen yang tidak dikategorikan benar/relevan oleh sistem sebagai kategori C_m, dan pada pengklasifikasian manual tidak berada pada kategori C_m.

2.11 Accuracy

Accuracy adalah persentase dari total *e-mail* yang benar diidentifikasi [DEF-10]. Pada pengklasifikasian ini, jumlah data yang benar dalam nilai a dijumlahkan dengan jumlah data bernilai d, kemudian dibagi total keseluruhan data pengujian. *Accuracy* dihitung dengan persamaan 2.8.

$$Accuracy = \left(\frac{a + d}{total\_data\_pengujian} \right) \text{x } 100 \qquad (2.8)$$

Keterangan :

- *Accuracy* = persentase nilai total dokumen lirik lagu yang benar diidentifikasi oleh sistem.
- d = menyatakan jumlah dokumen yang tidak dikategorikan benar/relevan oleh sistem sebagai kategori C_m, dan pada pengklasifikasian manual tidak berada pada kategori C_m.
- a = menyatakan jumlah dokumen yang dikategorikan sebagai kategori C_m oleh sistem yang juga merupakan hasil pengklasifikasian manual dengan kategori C_m.
- Total data pengujian = jumlah semua dokumen lirik lagu yang dijadikan dokumen *testing* dalam pengujian sistem.

BAB III
METODOLOGI PENELITIAN DAN PERANCANGAN

Pada bab metodologi peneltian dan perancangan ini akan dibahas langkah-langkah metode perancangan yang akan digunakan dalam skripsi ini. Langkah-langkahnya adalah :

1. Melakukan pencarian dan mempelajari studi literatur mengenai permasalahan klasifikasi emosi berdasarkan lirik lagu menggunakan metode algoritma *Iterative Dichotomizer Tree* (ID3) berasal dari buku maupun dari internet.
2. Persiapan data lirik lagu yang dibagi dalam dua bagian yaitu data *training* dan data *testing*.
3. Mengimplementasikan metode klasifikasi emosi berdasarkan lirik lagu dengan *Iterative Dichotomizer Tree* menggunakan data *training* yang telah disiapkan sebagai proses pelatihan, hasil pelatihan akan digunakan sebagai pengenalan ke sebuah sistem perangkat lunak.
4. Melakukan uji coba data *testing* terhadap hasil klasifikasi yang dilakukan oleh sistem. Hasil klasifikasi merupakan informasi pengklasifikasian dari emosi berdasarkan lirik lagu yang diujikan (data *testing*).
5. Melakukan evaluasi dan menganalisis hasil klasifikasi yang dihasilkan oleh sistem berupa tingkat akurasi.

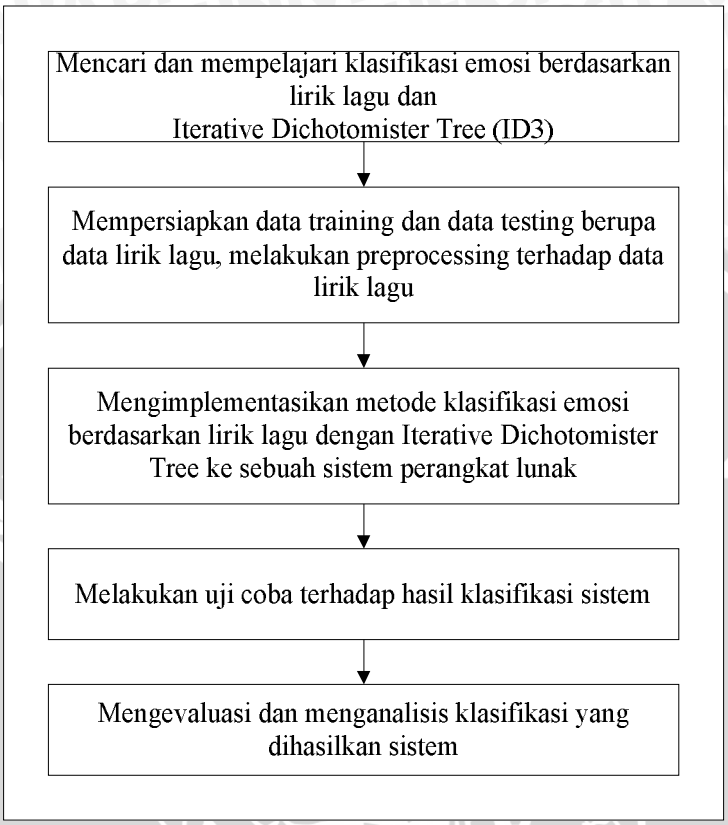
Langkah-langkah yang dilakukan dapat ditunjukkan pada gambar 3.1.

3.1 Analisa Data

Analisa data berisi pembahasan sumber data berupa dokumen lirik lagu yang diambil dari *website www.allmusic.com, www.popculturemadness.com*, serta *www.stereomood.com* dengan pengelompokkan emosi secara manual. Pembagian data untuk dokumen *training* dan *testing* berdasarkan perbandingan dalam pengujian data.

Pada situs tersebut nanti akan menjadi acuan untuk dokumen pembelajaran sistem. Untuk penelitian ini digunakan empat kategori emosi, kategori emosi yang digunakan adalah *angry songs, love songs, fun songs, sad songs*. Diambilnya keempat kategori ini untuk memperoleh data latih yang tepat, memaksimalkan

proses pembelajaran, dan meningkatkan nilai akurasi pada hasil pengujian dokumen.



Gambar 3.1 Alur penelitian

Dokumen lirik lagu dari situs *www.allmusic.com, www.popculturemadness.com,* dan *www.stereomood.com* dibagi menjadi dua bagian, bagian pertama digunakan sebagai dokumen lirik lagu pembelajaran yang disebut data *training* dan bagian kedua digunakan dokumen lirik lagu pengujian yang disebut data *testing*. Dalam pengujian data dilakukan perbandingan data *training* dan data *testing* untuk mendapatkan nilai keakurasian yang maksimal. Pembahasan untuk perbandingan data dijelaskan pada subbab pengujian data.

3.2 Deskripsi Umum Sistem

Secara keseluruhan penelitian ini dibagi dalam 3 tahap, yaitu *preprocessing*, pembelajaran dan klasifikasi.

Tahap pertama adalah *preprocessing*, semua data *input* berupa dokumen lirik lagu data *training* dan *testing* akan melalui *preprocessing*. Pada proses

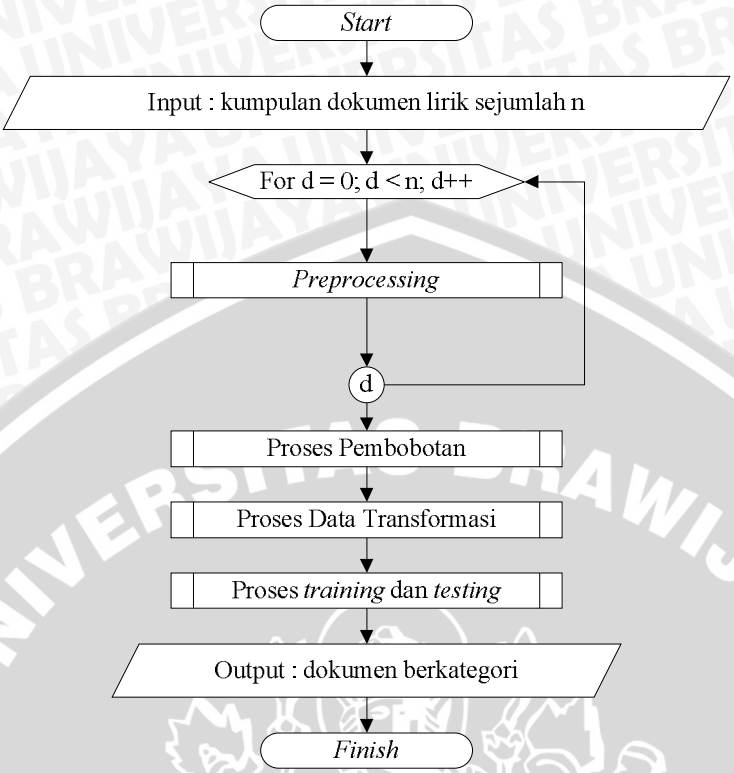
preprocessing terdapat beberapa langkah yakni *filtering, tokenizing, stemming*, dan frekuensi kata.

Tahap kedua adalah tahap pembelajaran atau *training*. Setelah melalui tahap *preprocessing*, data yang telah disiapkan akan melalui proses pembelajaran untuk membentuk *decision tree* dengan algoritma *Iterative Dichotomizer Tree.*

Tahap ketiga adalah tahap klasifikasi. Setelah sistem melalu proses pembelajaran dari data *training*, akan terbentuk suatu aturan-aturan atau *rule* hasil dari *decision tree*. Aturan tersebut nantinya digunakan pada data *testing* untuk mengetahui kelasnya.

Tahapan dalam diskripsi umum sistem, berdasarkan diagram pada gambar 3.2 adalah :

1. *User* memasukan data lirik dengan tipe **.txt,* data *training* dan data *testing* dimasukkan secara bersamaan pada tahap ini.
2. Setelah isi lirik berupa *text* diambil, akan melalui tahap *preprocessing*. Pada tahap ini akan dilakukan *tokenizing, stemming* dan stopword.
3. Dari hasil *preprocessing*, setiap kata akan dihitung bobotnya dengan metode perhitungan bobot TF-IDF.
4. Data hasil perhitungan TF-IDF dilakukan proses trasnformasi data.
5. Dalam proses ini terbagi menjadi dua, bila data *training* akan dilakukan proses pembelajaran. Bila data *testing* akan dilakukan proses pengklasifikasian berdasarkan hasil pengetahuan dari proses pembelajaran.
6. Hasil data berupa dokumen yang sudah berada di dalam kategori yang sudah ditentukan.



Gambar 3.2 *Flowchart* deskripsi umum *system*

3.3 Perancangan Proses

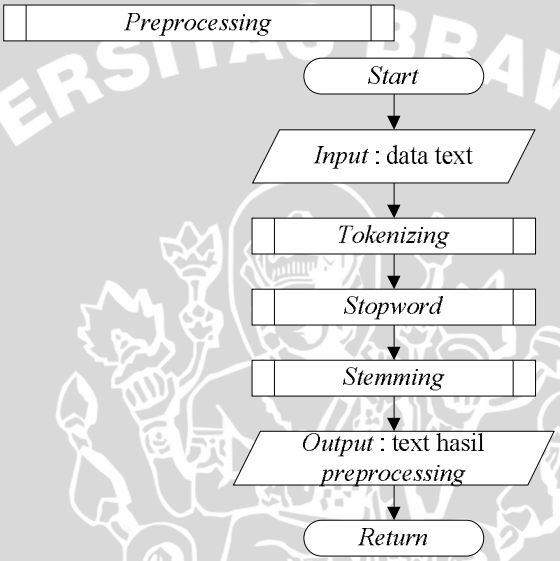
Pada sub bab perancangan proses akan dijelaskan mengenai proses-proses dalam membangun sistem. Seluruh dokumen lirik akan melalui proses ini baik data latih ataupun data uji. Pada sub bab ini akan menjelaskan *preprocessing*, pembobotan, *generalization*, pembelajaran dan klasifikasi.

3.3.1 Perancangan *Preprocessing*

Pada sub bab 3.3.1 dijelaskan perancangan *preprocessing*, pada tahapan ini dilakukan proses pemecahan dokumen (*tokenizing*), penghapusan kata yang kurang bermakna (*stopword*), pencarian kata dasar (*stemming*) dan penghitungan kemunculan kata pada tiap dokumen (frekuensi). Proses *preprocessing* digambarkan pada gambar 3.3.

Tahap *preprocessing* terbagi dalam tiga proses, berikut alur pemrosesan lirik pada *flowchart* perancangan *preprocessing*.

1. Lirik pertama kali akan dilakukan *tokenizing*, untuk penghapusan semua angka dan tanda baca.
2. Hasil *tokenizing*, selanjutnya melalui proses *stopword* penghapusan kata - kata yang kurang memiliki makna.
3. Selanjutnya lirik melalui proses *stemming*, proses pencarian kata dasar dari suatu kata berimbuhan.
4. Hasil dari *preprocessing* berupa kata dalam bentuk dasar dan tidak ada keterkaitan dengan kata - kata lainya.

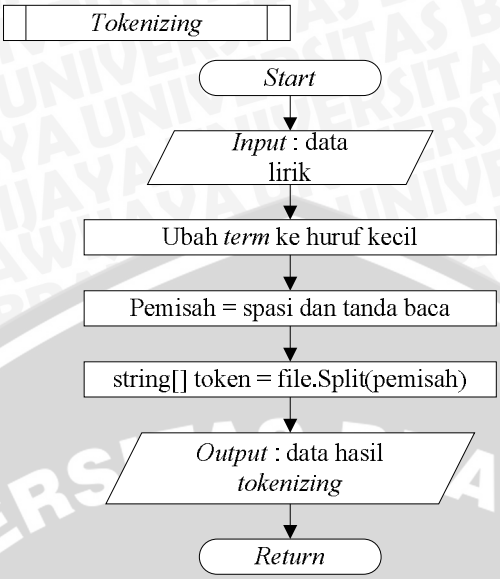


Gambar 3.3 *Flowchart* perancangan *preprocessing*

3.3.1.1 Perancangan Proses *Tokenizing*

Pada tahap *tokenizing* terdapat dua sub proses, *case folding* dan *parsing*. Perancangan proses *tokenizing* digambarkan pada gambar 3.4. Tahap *tokenizing* terbagi dalam dua tahap, *case folding* dan *parsing*. Berikut penjelasan untuk proses *tokenizing* pada gambar 3.4.

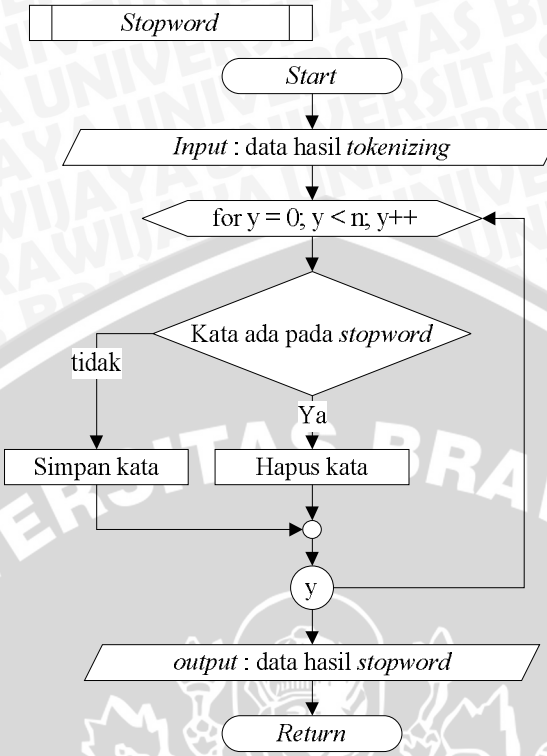
1. Data lirik akan di proses dalam proses *tokenizing*.
2. Data lirik akan dirubah kebentuk huruf kecil.
3. *Variable* pemisah mengunakan semua tanda baca, spasi dan karakter bukan huruf.
4. *String* akan dipisah sesuai dengan *variable* pemisah.
5. Hasil data pemisahan berupa data lirik yang terpisah.



Gambar 3.4 Flowchart proses tokenizing

3.3.1.2 Perancangan Proses *Stopword*

Perancangan proses *stopword* adalah dilakukannya pembuangan kata yang sering keluar tapi kurang memiliki informasi untuk mempresentasikan suatu dokumen. Data *stopword* terlampir pada lampiran 1, setiap kata akan dilakukan pencocokan dengan data *stopword* bila ada maka kata tersebut akan di hapus dan bila tidak maka kata tersebut akan di simpan. Secara keseluruhan perancangan proses *stopword* terdapat pada gambar 3.5.



Gambar 3.5 *Flowchart* proses *stopword*

Tahap *stopword*, dengan melakukan pengecekan pada *list stopword* yang sudah ada. Berikut perincian dari proses *stopword* pada gambar 3.5.

1. Data berupa kumpulan kata hasil *tokenizing* dimasukkan dalam sistem.
2. Satu persatu kata melalui proses pengecekan, jika kata ada dalam *list stopword* dilakukan penghapusan kata.
3. Kata tidak ditemukan dalam *stopword*, dilakukan penyimpanan kata tersebut.
4. Hasil dari sistem adalah kata yang tidak termasuk dalam *list stopword* dalam dokumen yang bersih dari *stopword*.

3.3.1.3 Perancangan Proses *Stemming*

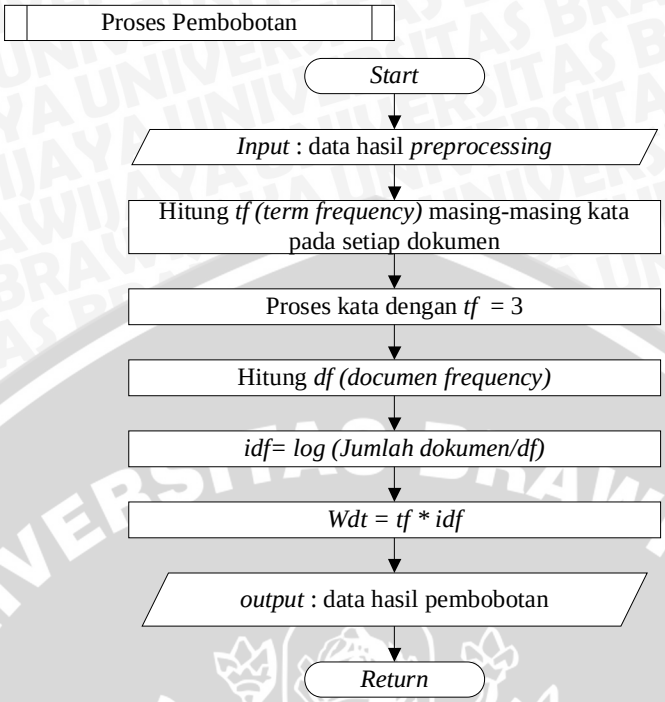
Pada proses *stemming* dilakukan pencarian kata dasar pada kata-kata berbahasa inggris menggunakan *porter stemmer*. Pada proses *porter stemmer* dihasilkan kumpulan kata yang sudah berbentuk kata dasar. Tahap pencarian kata dasar dapat dilihat pada *subbab* 2.4.2.

3.3.2 Perancangan Proses *tf-idf*

Proses selanjutnya setelah melalui pencarian kata dasar, melakukan pembobotan kata. Pembobotan kata adalah proses pencarian bobot suatu kata dari satu dokumen ke dokumen lainnya. Dalam proses pembobotan ini menggunakan metode *term frequency - inverse document frequency* yang sering dikenal dengan metode pembobotan *tf-idf*. Proses pembobotan kata dapat dilihat dari gambar 3.6.

Pada gambar 3.6 dijelaskan proses pembobotan menggunakan metode *tf-idf*, berikut penjelasan proses pembobotan.

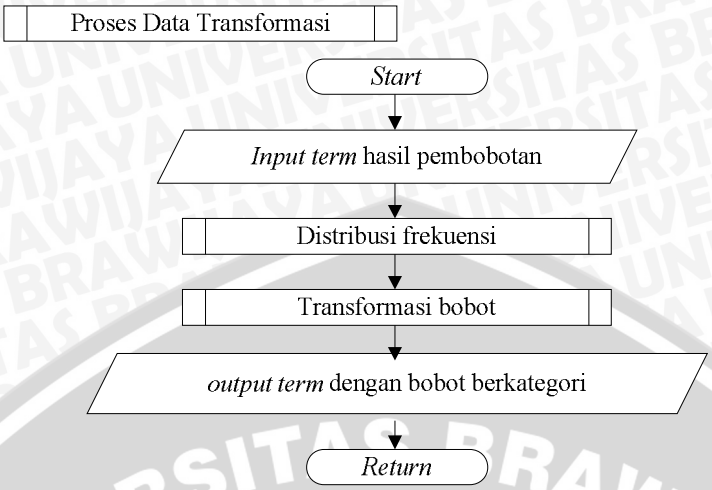
1. User memasukan kumpulan kata dalam suatu dokumen lirik.
2. Dilakukan perhitungan banyaknya kemunculan suatu kata dalam satu dokumen.
3. Dilakukan proses *feature selection*, kata yang akan diproses selanjutnya adalah kata yang memiliki frekuensi kemunculan sama dengan 3 dalam satu dokumen.
4. Selanjutnya dilakukan perhitungan jumlah kata yang muncul dalam suatu dokumen disebut juga perhitungan *df* atau *dokumen frequency*.
5. Setelah diketahui hasil *df*, proses berlanjut pada perhitungan *inverse* dokumen *frequency* dengan menggunakan persamaan 2.2.
6. Diketahui hasil *idf* dari setiap dokumen, proses selanjutnya menggunakan persamaan 2.1, dengan mengalikan setiap kemunculan kata dengan hasil *idf* dari tiap dokumen.
7. *Output* hasil dari pembobotan berupa atribut dan bobot atribut.



Gambar 3.6 *Flowchart* proses pembobotan

3.3.3 Perancangan Transformasi Data

Pada sub bab perancangan transformasi data, dilakukan perubahan dari data numerik menjadi data berkategori. Dilakukan transformasi data dengan cara seperti ini karena ID3 tidak dapat memproses data dengan tipe numerik, maka data pembobotan yang bertipe numerik akan diletakan dalam kelompok-kelompok dengan *range* data tertentu. Terdapat dua proses dalam perancangan transformasi data, distribusi frekuensi yang merupakan metode untuk pembuatan *range* dari data pembobotan dan pengkategorian bobot merupakan proses pemberian kategori pada bobot sesuai hasil *range* kategori yang dihasilkan dari distribusi frekuensi. Untuk perancangan proses data transformasi digambarkan pada gambar 3.7.

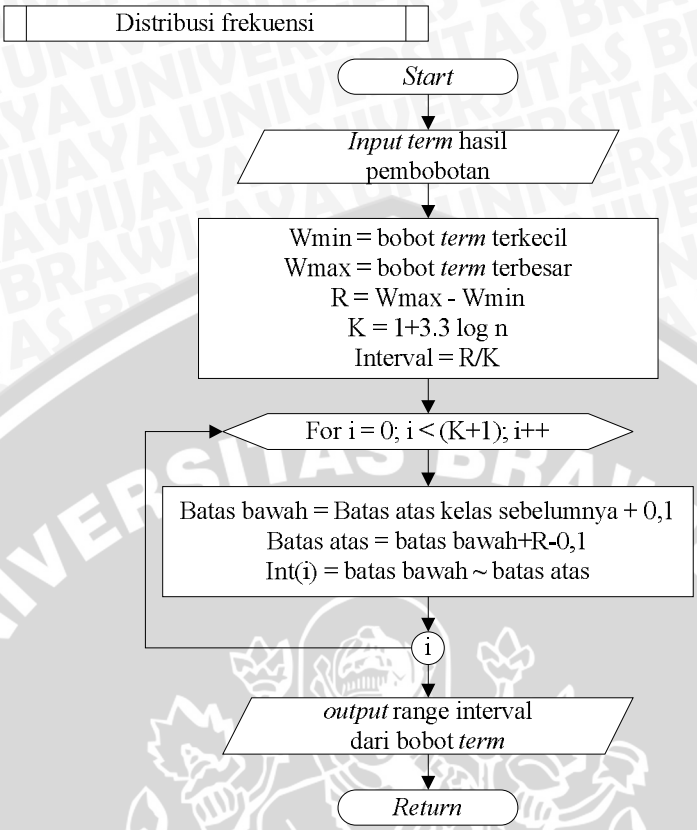


Gambar 3.7 *Flowchart* proses data transformasi

Untuk sub proses pertama pada proses data transformasi adalah distribusi frekuensi. Metode distribusi frekuensi digunakan untuk menentukan pembuatan pengelompokan data berdasarkan *range* data pada sekumpulan data.

Dari gambar 3.8 tentang pembuatan distribusi frekuensi, berikut tahapan pembuatan interval kelas dengan metode distribusi frekuensi.

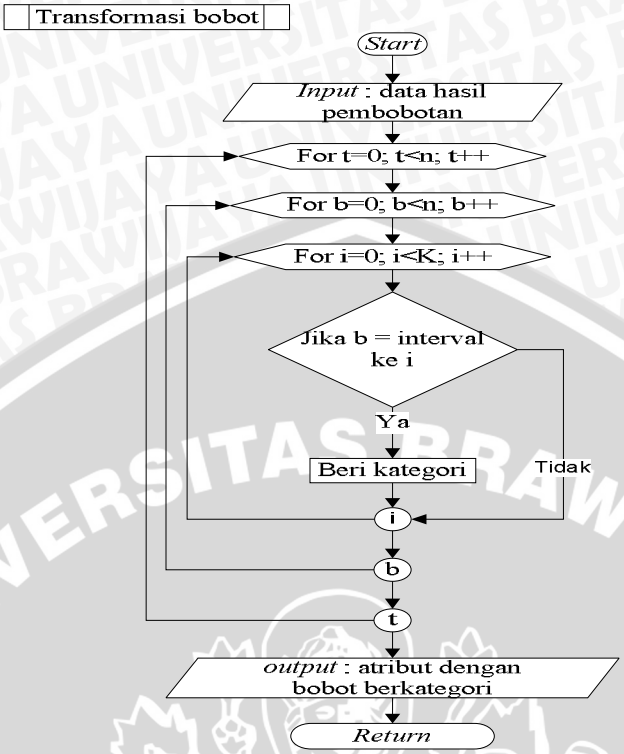
1. Diambil data *term* hasil pembobotan.
2. Di cari rentang dari bobot dengan cara mencari bobot *minimum* dan bobot maksimal terlebih dahulu, sehingga rentang dapat dihitung dengan persamaan 2.3.
3. Dicari banyaknya kelas interval dengan persamaan 2.4.
4. Setelah rentang dan kelas dihitung, maka dapat dicari nilai interval kelasnya dengan persamaan 2.5.
5. Dilakukan pembentukan interval tiap kelas.
6. Ditentukan batas bawah kelas yakni batas atas kelas sebelumnya ditambahkan 0,1, untuk kelas pertama menggunakan nilai bobot *minimum*.
7. Ditentukan batas atas kelas dengan cara menambahkan batas bawah kelas dengan rentang dan dikurangi dengan 0,1.
8. Setalah diketahui batas bawah dan batas atas kelas, maka dapat dibuat nilai interval ke-*n* adalah batas bawah sampai dengan batas atas.
9. Dihasilkan kelas interval beserta *range* tiap kelas interval.



Gambar 3.8 *Flowchart* proses distribusi frekuensi

Setelah dilakukan pembuatan interval setiap kelas dengan metode distribusi frekuensi, selanjutnya dilakukan proses transformasi data. Pada proses ini data akan dimasukan kedalam interval *range* sesuai dengan hasil distribusi frekuensi. Berikut proses transformasi data yang ditunjukan pada gambar 3.9.

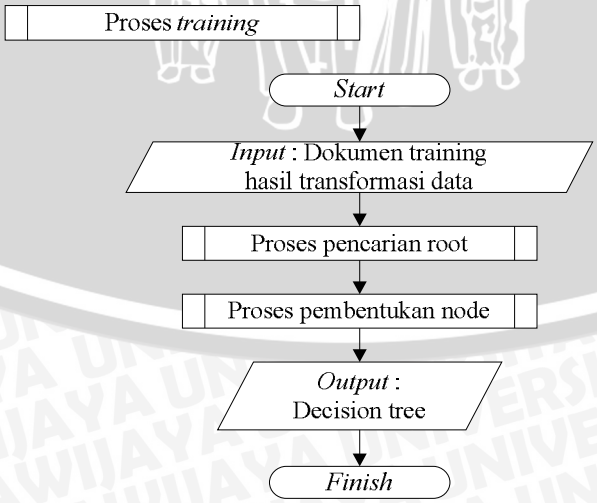
1. Setelah melalui proses pembuatan *range* dan interval setiap bobot, selanjutnya bobot akan dikategorikan.
2. Data Hasil pembobotan masuk kedalam proses pengkategorian bobot kata.
3. Proses pembobotan dilakukan perkata, sampai sejumlah kata yang ada.
4. Dilakukan pengecekan pada bobot kata.
5. Dari bobot ke-i akan di cocokan dengan interval yang telah dibentuk sebelumnya.
6. Jika bobot kata berada dalam satu interval ke-i maka akan di beri kategori, jika tidak berlanjut ke interval selanjutnya sampai menemukan bobot interval.
7. Data hasil proses transformasi disimpan dalam tipe *string*.



Gambar 3.9 *Flowchart* proses transformasi bobot

3.3.4 Pembelajaran

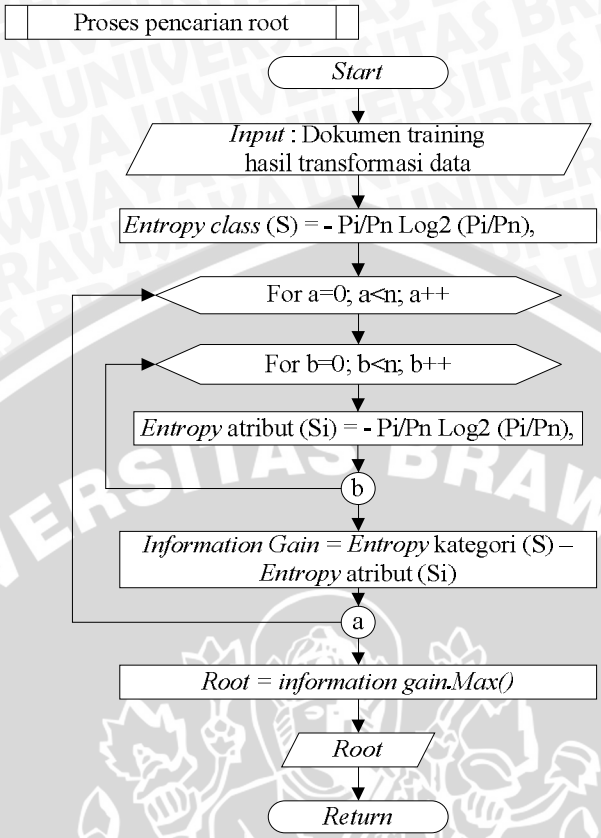
Pada proses pembelajaran dilakukan pembentukan pohon keputusan dengan algoritma ID3. Pada proses pembelajaran terdapat dua proses yakni proses pencarian *root* dan proses pembentukan *node* sebelum akhirnya nanti terbentuk pohon keputusan. Tahap proses *training* dapat dilihat pada gambar 3.10.



Gambar 3.10 *Flowchart* proses *training* ID3

Pada tahap pertama dijelaskan proses penentuan *root* dari perbandingan *entropy class* dengan *entropy* atribut. Penjelasan proses pencarian *root* dijelaskan pada gambar 3.11 proses pencarian *root*. Berikut penjelasan proses penentuan *root* dari *decision tree.*

1. *Input* data adalah dokumen *training* hasil transformasi data.
2. Data inputan berisi *term* dengan bobot sudah dalam kategori *range* sesuai hasil distribusi frekuensi.
3. Dilakukan perhitungan *entropy* class dengan persamaan 2.6.
4. Perulangan pertama dilakukan perhitungan *entropy* dari sejumlah data bobot berkategori yang berada pada *term* sejumlah n.
5. Perulangan kedua dilakukan perhitungan *entropy* bobot pada suatu *term*, sampai pada bobot sejumlah n pada suatu *term*.
6. Dilakukan perhitungan *information gain* dengan persamaan 2.7, dilakukan perbandingan antara *entropy* class dengan *entropy term* hasil perhitungan pada poin 5.
7. Setelah di dapat hasil *information gain* setiap *term*, kemudian dilakukan perbandingan dengan *term* yang lain, jika *term* tersebut memiliki IG tertinggi maka *term* atribut tersebut sebagai atribut terbaik dan digunakan sebagai *root*.
8. Hasil proses penentuan ini berupa *root*, cabang *root* nantinya akan digunakan sebagain acuan dalam proses pencarian *node* pada proses selanjutnya.

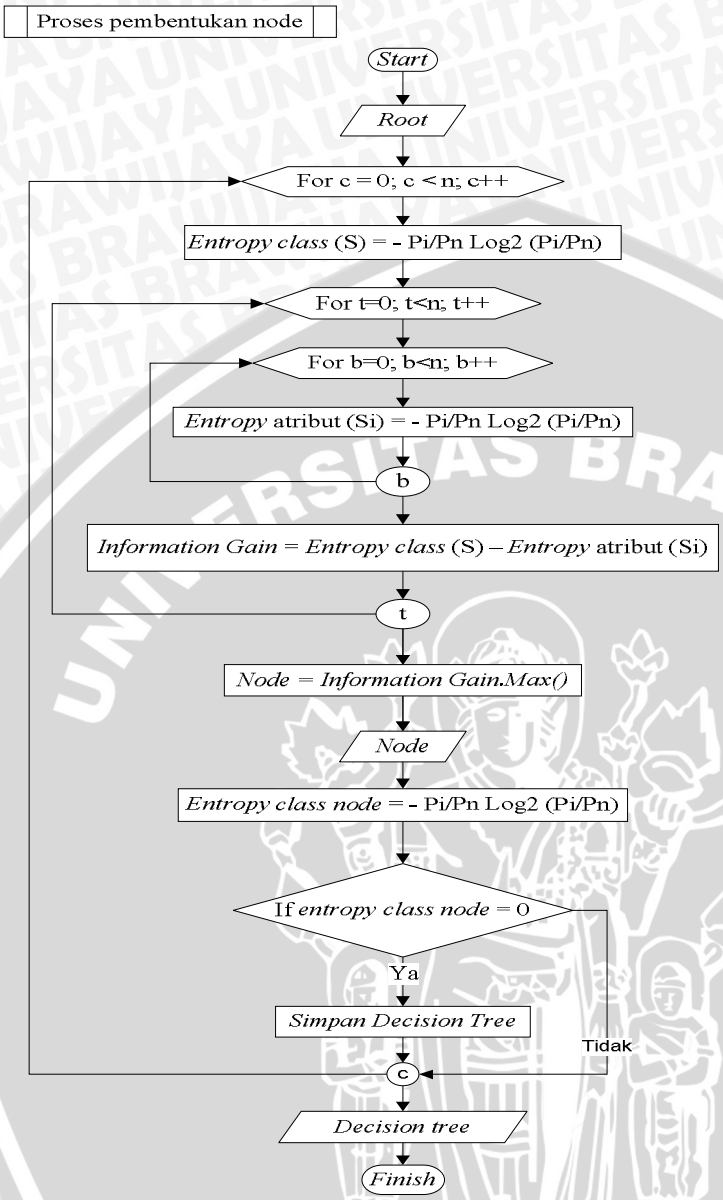


Gambar 3.11 *Flowchart* proses pencarian *root*

Pada tahap selanjutnya adalah tahap pembentukan *node*, dari hasil penentuan *root* pada proses sebelumnya selanjutnya cabang setiap *root* akan dilakukan perhitungan *entropy* dan perbandingan *information gain* sampai pada cabang terdapat satu *class* yang sama atau hasil *entropy* dari *node* bernilai nol. Berikut penjelasan proses pembentukan *node* yang digambarkan pada gambar 3.12 proses pembentukan *node* .

1. Dari hasil *root* yang dihasilkan pada proses sebelumnya, kemudian dilakukan perhitungan *entropy class* setiap cabang *root*.
2. Dilakukan pengecekan pada cabang *root*, dan dilakukan perulangan pada setiap cabang *root*.
3. Pada tiap cabang akan dilakukan perhitungan masing – masing *entropy class* pada tiap cabang dari *root*.
4. Lakukan perulangan sejumlah *term* atribut yang ada pada cabang.

5. Perulangan kedua dimana setiap *term* memiliki bobot kategori, dilakukan perulangan sampai sejumlah bobot kategori yang ada pada *term* atribut.
6. Dilakukan perhitungan *entropy* pada atribut.
7. Kembali ke perulangan kedua pada poin 5, sampai bobot pada *term* atribut tersebut habis.
8. Dilakukan perhitungan *information gain* dengan membandingkan *entropy class* dengan *entropy* atribut.
9. Setelah didapat *information gain*, diambil nilai terbesar dari *information gain* tersebut untuk digunakan sebagai *node*.
10. Dari hasil *node* dilakukan pengecekan *entropy class* yang berada pada cabang tersebut.
11. Jika *entropy class* bernilai nol, maka cabang tersebut merupakan akhir dari *tree*. Jika tidak maka kembali ke perulangan pada poin 2 (perulangan cabang *root*).
12. Hasil dari proses ini berupa pohon keputusan dari proses pembelajaran.



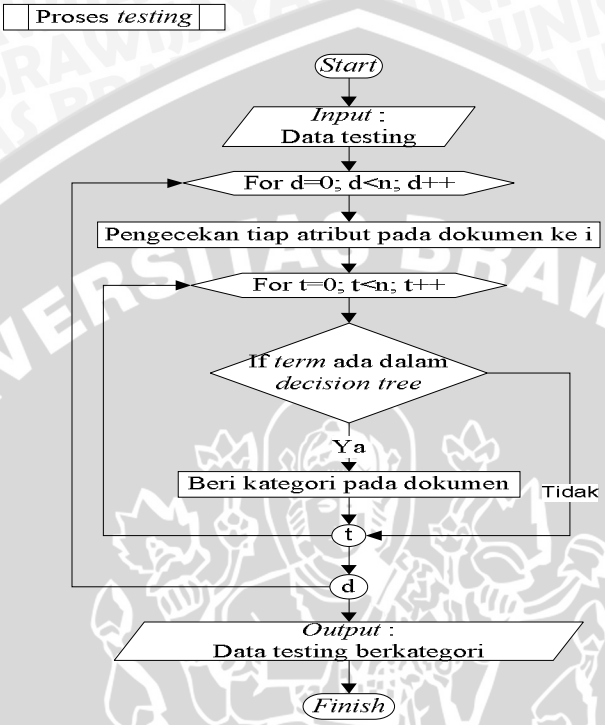
Gambar 3.12 *Flowchart* proses pembentukan *node*

3.3.5 Proses Klasifikasi

Pada proses *testing* dilakukan pengklasifikasian. Dokumen *testing* yang telah disiapkan kedalam sistem akan dilakukan pengklasifikasian sesuai hasil dari pembelajaran sistem. Hasil dari proses klasifikasi berupa dokumen yang sudah memiliki kategori, berikut penjelasan proses pengklasifikasian.

1. Proses dimulai dari memasukan kumpulan dokumen *testing* kedalam sistem.
2. Satu persatu dilakukan proses pengkategorian dokumen *testing*.

3. Proses dimulai dari bobot *term* pertama, jika *term* pertama memiliki kategori maka beri kategori pada dokumen *testing* tersebut. Jika tidak akan berulang pada *term* selanjutnya dari dokumen *testing* tersebut.
4. Hasil dari proses klasifikasi adalah dokumen lirik yang memiliki kategori.



Gambar 3.13 *Flowchart* proses klasifikasi ID3

3.4 Perhitungan Manual

Pada sub bab perhitungan manual akan dibahas contoh perhitungan manual untuk data *training*, *testing*, pembobotan *tf-idf*, proses generalisasi dengan distribusi frekuensi dan pengklasifikasian dengan algoritma ID3. Perhitungan manual dimulai setelah kata melalui tahap *preprocessing.*

3.4.1 Sumber Data

Data dalam perhitungan manual berjumlah 12 dokumen lirik lagu bahasa inggris. 12 dokumen lirik lagu bahasa inggris tersebut dibagi menjadi dua bagian, 8 dokumen lirik lagu bahasa inggris sudah memiliki kategori yang nantinya akan di gunakan sebagai contoh perhitungan dokumen *training*. 4 dokumen lirik lagu bahasa inggris sisanya tidak berkategori dan digunakan sebagai data *testing.* 8

dokumen *training* terbagi menjadi empat kategori, yaitu kategori *angry, love, fun, dan sad*. Tiap kategori terdiri dari dua dokumen. Dokumen dalam perhitungan manual dan perhitungan jumlah frekuensi kata ada pada lampiran 2.

3.4.2 Proses Pembobotan

Setelah melakukan perhitungan frekuensi setiap kata pada 12 dokumen, kemudian dilakukan *stopword* dan *stemming* secara manual. Data yang diambil dalam proses selanjutnya adalah kata yang memiliki frekuensi > 2. Daftar kata yang dihasilkan dari tahap *preprocessing* terdapat dalam dua tabel berikut, tabel 3.1 hasil *preprocessing* dari data *training* dan tabel 3.2 hasil *preprocessing* untuk data *testing*. Kedua tabel yang berisi kata tersebut akan dilakukan proses pembobotan. Berikut hasil perhitungan *term frequency, document frequency* dan *invers document frequency*, perhitungan *idf* dengan menggunakan persamaan (2.2). Berikut contoh perhitungan *idf* untuk kata *screw*, dan dengan jumlah dokumen sebanyak 8 untuk data *training* dan 4 untuk data *testing.*

$$idf = \log \frac{8}{2} = 0{,}60206$$

Tabel 3.1 Data *training* untuk *term frequency*, *document frequency* dan *invers document frequency*

TF	D1	D2	D3	D4	D5	D6	D7	D8	DF	IDF
screw	**0**	**0**	**0**	**0**	**25**	**10**	**0**	**0**	**2**	**0.60206**
english	4	0	0	0	0	0	0	0	1	0.90309
adverb	3	0	0	0	0	0	0	0	1	0.90309
don't	5	0	0	0	0	0	0	0	1	0.90309
word	8	0	0	0	0	0	0	0	1	0.90309
shirlei	5	0	0	0	0	0	0	0	1	0.90309
hair	0	3	0	0	0	0	0	0	1	0.90309
man	0	0	3	0	0	0	0	0	1	0.90309
kid	0	0	4	0	0	0	0	0	1	0.90309
better	0	0	4	0	0	0	0	0	1	0.90309
don	0	0	9	0	0	3	0	0	2	0.60206
time	0	0	0	3	0	0	0	0	1	0.90309
grade	0	0	0	3	0	0	0	0	1	0.90309
mood	0	0	0	0	6	0	0	0	1	0.90309
love	0	0	0	0	8	7	0	0	2	0.60206
babi	0	0	0	0	0	3	7	0	2	0.60206
live	0	0	0	0	0	0	9	0	1	0.90309
life	0	0	0	0	0	0	5	0	1	0.90309
leav	0	0	0	0	0	0	3	0	1	0.90309
breath	0	0	0	0	0	0	3	0	1	0.90309
surviv	0	0	0	0	0	0	3	0	1	0.90309
hurt	0	0	0	0	0	0	0	4	1	0.90309
will	0	0	0	0	0	0	0	4	1	0.90309

Tabel 3.2 Data *testing* untuk *term frequency*, *document frequency* dan *invers document frequnecy*

TF	D9	D10	D11	D12	DF	IDF
awak	3	0	0	0	1	0.90309
feel	3	0	0	0	1	0.90309
hate	10	0	0	0	1	0.90309
dai	0	5	0	0	1	0.90309
smile	0	3	0	0	1	0.90309
face	0	3	0	0	1	0.90309
number	0	4	11	0	2	0.60206
hei	0	0	5	0	1	0.90309
met	0	0	5	0	1	0.90309
crazi	0	0	5	0	1	0.90309
call	0	0	13	0	1	0.90309
hard	0	0	3	0	1	0.90309
boi	12	0	3	0	2	0.60206
chase	0	7	3	0	2	0.60206
miss	0	0	9	0	1	0.90309
bad	0	0	11	0	1	0.90309
runnin	0	0	0	4	1	0.90309
round	0	0	0	4	1	0.90309
scar	0	0	0	4	1	0.90309
collect	0	0	0	4	1	0.90309
jar	0	0	0	4	1	0.90309
heart	0	0	0	4	1	0.90309
tear	0	0	0	4	1	0.90309
apart	0	0	0	4	1	0.90309
gonna	0	0	0	4	1	0.90309
catch	0	0	0	4	1	0.90309
cold	0	0	0	4	1	0.90309
ic	0	0	0	4	1	0.90309
insid	0	0	0	4	1	0.90309
soul	0	0	0	4	1	0.90309
don	0	0	0	8	1	0.90309
time	0	4	0	3	2	0.60206
love	8	3	0	6	3	0.425969
babi	0	0	5	0	1	0.90309
life	0	0	4	0	1	0.90309
leav	0	0	0	4	1	0.90309

Setelah diketahui hasil *idf* selanjutnya dilakukan pembobotan kata pada setiap dokumen dengan menggunakan persamaan 2.1. Berikut contoh perhitungan pembobotan untuk kata *screw* pada dokumen ke-1. Tabel hasil pembobotan berada pada tabel 3.3 untuk data *training*.

$$wdt = 25 \times 0.60206 = 15.051$$

Tabel 3.3 Data *training* hasil pembobotan kata

WDT	D1	D2	D3	D4	D5	D6	D7	D8
screw	**0**	**0**	**0**	**0**	**15.051**	**6.021**	**0**	**0**
english	3.612	0	0	0	0	0	0	0
adverb	2.709	0	0	0	0	0	0	0
don't	4.515	0	0	0	0	0	0	0
word	7.224	0	0	0	0	0	0	0
shirlei	4.515	0	0	0	0	0	0	0
hair	0	2.709	0	0	0	0	0	0
man	0	0	2.709	0	0	0	0	0
kid	0	0	3.612	0	0	0	0	0
better	0	0	3.612	0	0	0	0	0
don	0	0	5.418	0	0	1.806	0	0
time	0	0	0	2.709	0	0	0	0
grade	0	0	0	2.709	0	0	0	0
mood	0	0	0	0	5.418	0	0	0
love	0	0	0	0	4.816	4.214	0	0
babi	0	0	0	0	0	1.806	4.214	0
live	0	0	0	0	0	0	8.127	0
life	0	0	0	0	0	0	4.515	0
leav	0	0	0	0	0	0	2.709	0
breath	0	0	0	0	0	0	2.709	0
surviv	0	0	0	0	0	0	2.709	0
hurt	0	0	0	0	0	0	0	3.612
will	0	0	0	0	0	0	0	3.612

Tabel 3.4 Data *testing* hasil pembobotan kata

WDT	D9	D10	D11	D12
awak	2.70927	0	0	0
feel	2.70927	0	0	0
hate	9.0309	0	0	0
dai	0	4.51545	0	0
smile	0	2.70927	0	0
face	0	2.70927	0	0
number	0	2.40824	6.62266	0
hei	0	0	4.51545	0
met	0	0	4.51545	0
crazi	0	0	4.51545	0
call	0	0	11.74017	0
hard	0	0	2.70927	0
boi	0	0	2.70927	0
chase	0	0	2.70927	0
miss	0	0	8.12781	0
bad	0	0	9.93399	0
runnin	0	0	0	3.61236
round	0	0	0	3.61236
scar	0	0	0	3.61236
collect	0	0	0	3.61236
insid	0	0	0	3.61236
soul	0	0	0	3.61236
don	0	0	0	7.22472
time	0	2.40824	0	1.80618
love	3.40775	1.277906	0	2.555812
babi	0	0	4.51545	0
life	0	0	3.61236	0
leav	0	0	0	3.61236

3.4.3 Proses *Data Transformation*

Pada proses *data transformation* data berupa numerik tidak dapat diterima oleh ID3, karena ID3 memproses data dengan menggunakan data bertipe kategori. Dalam masalah ini digunakan data trasformasi *generalization*, dimana pengkategorian data dalam kelas–kelas tertentu dengan metode distribusi frekuensi. Dalam penerapan kedalam sistem distribusi frekuensi dilakukan pada setiap atribut karena setiap atribut memiliki data bobot berbeda. Pada perhitungan manual ini distribusi frekuensi tidak dilakukan ke seluruh atribut karena jumlah dokumen terbatas, maka dilakukan distribusi frekuensi keseluruh data hasil pembobotan dari data *training*. Berikut langkah-langkah untuk membuat distribusi frekuensi.

1. Hasil data pembobotan *training* dilakukan pengurutan data dan mencari data terkecil dan data terbesar (X_{min} dan X_{max}). Dilakukan pencarian *range* antara kedua data tersebut dengan persamaan 2.3. Diketahui nilai maksimal dari data adalah 15.0515 dan nilai *minimum* dari data adalah 0 dengan jumlah data sampel sejumlah 184. Dilakukan perhitungan rentang nilai *minimum* dan maksimal dengan persamaan 2.3.

$$R = 15.0515 - 0 = 15.0515$$

2. Menentukan banyak kelas yang akan digunakan dengan menggunakan pesamaan 2.4.

$$K = 1 + 3.3\log 184 = 8.474$$

Dari hasil perhitungan kelas diatas maka kelas yang dapat dibuat dari data tersebut sebanyak **8 kelas**.

3. Menentukan panjang interval setiap kelas dengan menggunakan persamaan 2.5.

$$\text{Interval} = \frac{15.0515}{8} = 1.88$$

Dari hasil perhitungan interval diatas, maka interval kelas dengan panjang **2**.

4. Memilih ujung bawah kelas interval pertama. Untuk ini, bisa diambil sama dengan data terkecil atau mulai data yang lebih kecil, tapi selisihnya harus kurang dari panjang kelasnya. Maka diambil 0 sebagai sampel data terkecil, menentukan batas kelas atas dengan menambahkan 0 dengan interval

kemudian mengurangi batas atas kelas dengan skala terkecil dari data. Skala terkecil data menggunakan 0.1. Maka didapat interval kelas pertama 0 – 1.9, dan interval kelas kedua 2 – 3.9 dan demikian seterusnya.

5. Membuat tabel distribusi frekuensi sesuai aturan sebelumnya. Setiap *range* interval memiliki nama untuk memudahkan pemanggilan *record* data. Untuk mengatasi data dengan nilai bobot lebih dari data X_{max} dalam distribusi frekuensi, maka pada data maksimal dilakukan perlakuan berbeda dengan menggunakan kondisi data x > max. Hasil distribusi frekuensi ditunjukan pada tabel 3.5.

Tabel 3.5 Tabel hasil pengurutan data pembobotan

Data interval	Nilai interval	Frekuensi
Int 1	0 – 1.9	159
Int 2	2 – 3.9	13
Int 3	4 – 5.9	8
Int 4	6 – 7.9	2
Int 5	8 – 9.9	1
Int 6	10 – 11.9	0
Int 7	12 – 13.9	0
Int 8	X > 14	1
Jumlah Data		184

Dari hasil tabel distirbusi frekuensi dihasilkan pengkategorian untuk pembobotan nilai data, selanjutnya melakukan trasformasi bobot pada tabel 3.3. Nilai setiap bobot akan digantikan dengan data interval sesuai dengan pengkategorian data pada tabel 3.5. Hasil transformasi data dan label kategori dari setiap dokumen ditunjukan pada tabel 3.6.

Tabel 3.6 Data *training* hasil transformasi data

Term	D1	D2	D3	D4	D5	D6	D7	D8
screw	int 1	int 1	int 1	int 1	int 8	int 4	int 1	int 1
english	int 2	int 1	int 1	int 1	int 1	int 1	int 1	int 1
adverb	int 2	int 1	int 1	int 1	int 1	int 1	int 1	int 1
don't	int 3	int 1	int 1	int 1	int 1	int 1	int 1	int 1
word	int 4	int 1	int 1	int 1	int 1	int 1	int 1	int 1
shirlei	int 3	int 1	int 1	int 1	int 1	int 1	int 1	int 1
hair	int 1	int 2	int 1	int 1	int 1	int 1	int 1	int 1
man	int 1	int 1	int 2	int 1	int 1	int 1	int 1	int 1
kid	int 1	int 1	int 2	int 1	int 1	int 1	int 1	int 1
better	int 1	int 1	int 2	int 1	int 1	int 1	int 1	int 1
don	int 1	int 1	int 3	int 1	int 1	int 1	int 1	int 1
time	int 1	int 1	int 1	int 2	int 1	int 1	int 1	int 1
grade	int 1	int 1	int 1	int 2	int 1	int 1	int 1	int 1
mood	int 1	int 1	int 1	int 1	int 3	int 1	int 1	int 1
love	int 1	int 1	int 1	int 1	int 3	int 3	int 1	int 1
babi	int 1	int 1	int 1	int 1	int 1	int 1	int 3	int 1
live	int 1	int 1	int 1	int 1	int 1	int 1	int 5	int 1
life	int 1	int 1	int 1	int 1	int 1	int 1	int 3	int 1
leav	int 1	int 1	int 1	int 1	int 1	int 1	int 2	int 1
Kategori	**angry**	**angry**	**fun**	**fun**	**love**	**love**	**sad**	**sad**

3.4.4 Perancangan Pohon Keputusan

Pada sub bab ini dibahas perhitungan dalam pembentukan pohon keputusan dan contoh pembuatan pohon keputusan. Dari tabel 3.6 yang dihasilkan pada proses trasformasi data, data sudah dalam bentuk kategori sehingga sudah bisa dilakukan perhitungan menggunakan persamaan ID3. Berikut langkah perhitungan pembentukan *tree*.

1. Pembentukan *root* utama.

Melakukan perbandingan *entropy* atribut kategori dengan atribut kata dengan menggunakan persamaan 2.6.

Menghitung *entropy* kategori (S)

S adalah 8 data dari tabel kategori pada tabel 3.6 dengan prediksi masing-masing tiap kategori *angry*, *fun*, *love*, dan *sad* 2 dokumen. Dengan menggunakan persamaan 2.6.

$$\text{Entropy kategori}(S) = \left(-\left(\frac{2}{8}\right)\log_2\left(\frac{2}{8}\right) \text{x } 4 \right) = 2$$

Menghitung *entropy* atribut dan menentukan *information gain*

Berikut contoh perhitungan *entropy* dan information *gain* untuk atribut *screw*, *information gain* dengan persamaan *Gain* (A) = Info (D) – Info_A (D) (2.7). Dengan jumlah data 8, *record* data int 8 dengan jumlah 1 (dalam kategori *angry* 1 sedangkan kategori lainnya 0), *record* data int 4 dengan jumlah data 1 (kategori *angry* 1 sedangkan kategori lainnya 0), *record* data int 1 dengan jumlah data 6 (kategori *angry* 0 sedangkan pada kategori *fun*, *love*, dan *sad* masing-masing 2).

Entropy screw int 8 1 (1,0,0,0), int 4 1 (1,0,0,0), dan int 1 6 (0,2,2,2)

$$\text{Entropy int } 8\,(1,0,0,0) = -\left(\frac{1}{1}\right)\log_2\left(\frac{1}{1}\right) - \left(\left(\frac{0}{1}\right)\log_2\left(\frac{0}{1}\right) \text{x } 3\right) = 0$$

$$\text{Entropy int } 4\,(1,0,0,0) = -\left(\frac{1}{1}\right)\log_2\left(\frac{1}{1}\right) - \left(\left(\frac{0}{1}\right)\log_2\left(\frac{0}{1}\right) \text{x } 3\right) = 0$$

$$\text{Entropy int } 1\,(0,2,2,2) = -\left(\frac{0}{6}\right)\log_2\left(\frac{0}{6}\right) - \left(\left(\frac{2}{6}\right)\log_2\left(\frac{2}{6}\right) \text{x } 3\right) = 1.585$$

Information Gain screw

Perbandingan *entropy* yang dimiliki *screw* dengan *entropy* pada kelas kategori.

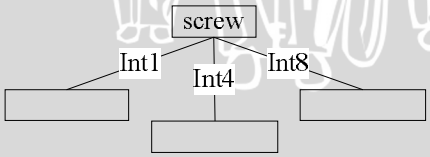
$$\text{IG screw} = 2 - \left(\frac{1}{8}\right)0 - \left(\frac{1}{8}\right)0 - \left(\frac{6}{8}\right)1.585 = 0.811$$

Berikut hasil perhitungan entropi dan *information gain* pada seluruh atribut ditampilkan pada tabel 3.7.

Tabel 3.7 Hasil perhitungan *entropy* dan *information gain*

Term	**Int1**	**Int2**	**Int3**	**Int4**	**Int5**	**Int6**	**Int7**	**Int8**	**IG**
screw	**1.585**	**x**	**x**	**0**	**x**	**x**	**x**	**0**	**0.811**
english	0.9709	0	x	x	x	x	x	x	0.1909
adverb	0.9709	0	x	x	x	x	x	x	0.1909
don't	1	x	x	x	x	x	x	x	0
word	1	x	x	x	x	x	x	x	0
shirlei	1	x	x	x	x	x	x	x	0
hair	1	x	x	x	x	x	x	x	0
man	1	x	x	x	x	x	x	x	0
kid	0.9709	0	x	x	x	x	x	x	0.1909
better	1	x	x	x	x	x	x	x	0
don	0.9709	x	0	x	x	x	x	x	0.1909
time	0.9709	0	x	x	x	x	x	x	0.1909
grade	0.8112	0	0	x	x	x	x	x	0.4592
mood	**0**	**0**	**x**	**x**	**x**	**x**	**x**	**x**	**1**
love	0.9709	x	0	x	x	x	x	x	0.1909
babi	1	x	x	x	x	x	x	x	0
live	0.9709	x	0	x	x	x	x	x	0.1909
life	0.9709	0	x	x	x	x	x	x	0.1909
leav	0.9709	0	x	x	x	x	x	x	0.1909
breath	**0.9709**	**x**	**x**	**x**	**x**	**x**	**x**	**x**	**0.1909**
surviv	0.9709	0	x	x	x	x	x	x	0.1909
hurt	0.9709	0	x	x	x	x	x	x	0.1909
will	1	x	x	x	x	x	x	x	0

Dari hasil perhitungan *information gain* pada tabel, atribut *screw* merupakan atribut terbaik karena memiliki IG terbesar dibandingkan IG atribut yang lainnya. Gambar *tree* yang terbentuk dari perhitungan *root* pada gambar 3.14.



Gambar 3.14 *Tree* hasil perhitungan *root*

2. *Node* cabang Int 1 dari atribut *screw.*

Dilakukan pengecekan atribut mana saja yang berada pada cabang Int 1, data atribut yang berada pada cabang Int 1 terdapat pada tabel 3.8.

Tabel 3.8 Data atribut pada cabang Int1 untuk atribut *screw*

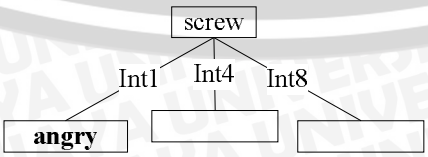
Term	**D1**	**D2**
english	int 2	int 1
adverb	int 2	int 1
don't	int 3	int 1
word	int 4	int 1
shirlei	int 3	int 1
hair	int 1	int 2
man	int 1	int 1
kid	int 1	int 1
better	int 1	int 1
don	int 1	int 1
time	int 1	int 1
grade	int 1	int 1
mood	int 1	int 1
love	int 1	int 1
babi	int 1	int 1
live	int 1	int 1
life	int 1	int 1
leav	int 1	int 1
breath	int 1	int 1
surviv	int 1	int 1
hurt	int 1	int 1
will	int 1	int 1
Kategori	**angry**	**angry**

Dari data tabel dilakukan perhitungan *entropy* untuk kelas kategori, jumlah dokumen 2 dengan 2 data pada kategori *angry*. Pada kategori *fun, love, dan sad* 0 data.

Entropy kategori 2 (2,0,0,0)

$$\text{Entropy kategori } (S_{\text{int1}}) = -\left(\frac{2}{2}\right)\log_2\left(\frac{2}{2}\right) - \left(\left(\frac{0}{2}\right)\log_2\left(\frac{0}{2}\right) \text{x } 3\right) = 0$$

Dari hasil *entropy* kelas diperoleh nilai 0 yang berarti sampel data yang berada pada cabang Int1 untuk atribut *screw* sudah dalam satu atribut yang sama, maka tidak dilakukan rekursif dan proses kembali ke *node* sebelumnya. Pada cabang Int 1 untuk atribut *screw* memiliki label *angry*. Hasil tree dari cabang ini ditunjukan pada gambar 3.15.

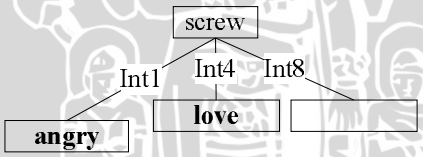


Gambar 3.15 *Tree* hasil cabang Int 1 dari atribut *screw*

2.1 Pada cabang Int4 pada atribut *screw* sudah dalam atribut target, maka tidak dilakukan rekursif dan label masing–masing cabang diberikan sesuai atribut yang tersedia pada atribut target. Gambar *tree* ditunjukan pada gambar 3.16.

Tabel 3.9 Data atribut pada cabang Int 4

Term	D6
english	int 1
adverb	int 1
don't	int 1
word	int 1
shirlei	int 1
hair	int 1
man	int 1
kid	int 1
better	int 1
don	int 1
time	int 1
grade	int 1
mood	int 3
love	int 3
babi	int 1
live	int 1
life	int 1
leav	int 1
breath	int 1
surviv	int 1
hurt	int 1
will	int 1
Kategori	**love**

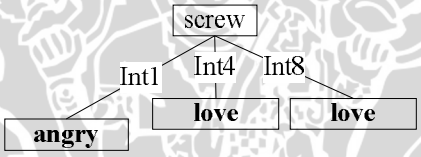


Gambar 3.16 *Tree* hasil cabang Int 4 dari atribut *screw*

2.2 Pada cabang Int 8 pada atribut *screw* sudah dalam atribut target, maka tidak dilakukan rekursif dan label masing–masing cabang diberikan sesuai atribut yang tersedia pada atribut target. Gambar *tree* ditunjukan pada gambar 3.17.

Tabel 3.10 Data atribut pada cabang Int 8

Term	D5
english	int 1
adverb	int 1
don't	int 1
word	int 1
shirlei	int 1
hair	int 1
man	int 1
kid	int 1
better	int 1
don	int 1
time	int 1
grade	int 1
mood	int 3
love	int 3
babi	int 1
live	int 1
life	int 1
leav	int 1
breath	int 1
surviv	int 1
hurt	int 1
will	int 1
Kategori	**love**



Gambar 3.17 *Tree* hasil cabang Int 8 dari atribut *screw*

Ketika semua data sampel sudah dalam satu atribut target maka rekursif sudah tidak berlanjut, ada gambar 3.17 merupakan hasil *tree* dari proses pembelajaran lirik lagu.

3.4.5 Perancangan *Rule Tree*

Setelah dihasilkan *decision tree* dari proses pembelajaran, dilakukan pengekstrakan hasil *decision tree* untuk mendapatkan aturan dalam pengklasifikasian dari *decision tree* tersebut. Berikut aturan klasifikasi yang didapat :

1. **IF** atribut *screw* = Int 1 **THEN** lirik lagu termasuk dalam kategori *angry*.
2. **IF** atribut *screw* = Int 4 **THEN** lirik lagu termasuk dalam kategori *love*.
3. **IF** atribut *screw* = Int 8 **THEN** lirik lagu termasuk dalam kategori *love*.

3.4.6 Perancangan Proses Klasifikasi

Pada sub bab sebelumnya hasil *decision tree* telah terbentuk dan telah dilakukan pengekstrakan untuk menghasilkan aturan dalam pengklasifikasian. Pada tahap ini dilakukan proses klasifikasi sesuai dengan hasil aturan klasfikasi yang terbentuk pada dokumen pembelajaran.

Pada data *testing* dilakukan proses yang sama dengan data *training*, sampai pada proses pembobotan. Setelah setiap atribut mendapatkan nilai bobot, dilakukan data transformasi. Pada data trasformasi tidak dilakukan perhitungan distribusi frekuensi seperti pada data *training*, data bobot yang ada pada data *testing* tinggal mengikuti hasil distribusi frekuensi dari hasil data *training*. Berikut hasil data trasformasi untuk data *testing* pada tabel 3.11.

Tabel 3.11 Data *testing* hasil generalisasi

Term	**D9**	**D10**	**D11**	**D12**
awak	int 2	int 1	int 1	int 1
feel	int 2	int 1	int 1	int 1
hate	int 5	int 1	int 1	int 1
dai	int 1	int 3	int 1	int 1
smile	int 1	int 2	int 1	int 1
face	int 1	int 2	int 1	int 1
number	int 1	int 2	int 4	int 1
hei	int 1	int 1	int 3	int 1
met	int 1	int 1	int 3	int 1
crazi	int 1	int 1	int 3	int 1
call	int 1	int 1	int 6	int 1
hard	int 1	int 1	int 2	int 1
boi	int 1	int 1	int 2	int 1
chase	int 1	int 1	int 2	int 1
miss	int 1	int 1	int 5	int 1
bad	int 1	int 1	int 5	int 1
runnin	int 1	int 1	int 1	int 2
round	int 1	int 1	int 1	int 2
scar	int 1	int 1	int 1	int 2
collect	int 1	int 1	int 1	int 2
jar	int 1	int 1	int 1	int 2
heart	int 1	int 1	int 1	int 2
tear	int 1	int 1	int 1	int 2
apart	int 1	int 1	int 1	int 2
gonna	int 1	int 1	int 1	int 2
catch	int 1	int 1	int 1	int 2
cold	int 1	int 1	int 1	int 2
Ic	int 1	int 1	int 1	int 2
insid	int 1	int 1	int 1	int 2
soul	int 1	int 1	int 1	int 2
don	int 1	int 1	int 1	int 4
time	int 1	int 2	int 1	int 1
love	int 1	int 1	int 1	int 1
babi	int 1	int 1	int 3	int 1
life	int 1	int 1	int 2	int 1
leav	int 1	int 1	int 1	int 2

Pada tabel 3.6 dilakukan pengklasifikasian terhadap data *testing* sesuai aturan klasifikasi yang sudah terbentuk. Langkah pengklasifikasian dilakukan perdokumen, dimulai dengan atribut yang memiliki nilai bobot tertinggi (pada distribusi frekuensi ini kelas data tertinggi adalah Int 8) proses berhenti ketika atribut tertinggi telah mendapatkan label kategori.

Pada data *testing* D9 dilakukan pengurutan nilai bobot, dimulai dari Int 5 atribut *hate*, Int 2 dari atribut *awak, feel*, Int 1 dari atribut *love, dai, smile, face, number, hei, met, crazi, call, hard, boi, chase, miss, bad, runnin, round, scar, collect, jar, heart, tear, apart, gonna, catch, cold, ic, insid, soul, don, time, babi, life,* dan *leav*. Dari dokumen lirik lagu D9 atribut yang bisa masuk dalam aturan klasifikasi adalah atribut *love* dengan int 1, aturan klasifikasi yang sesuai untuk kondisi D9 adalah aturan ke satu maka dokumen lirik lagu D9 berlabel *angry*.

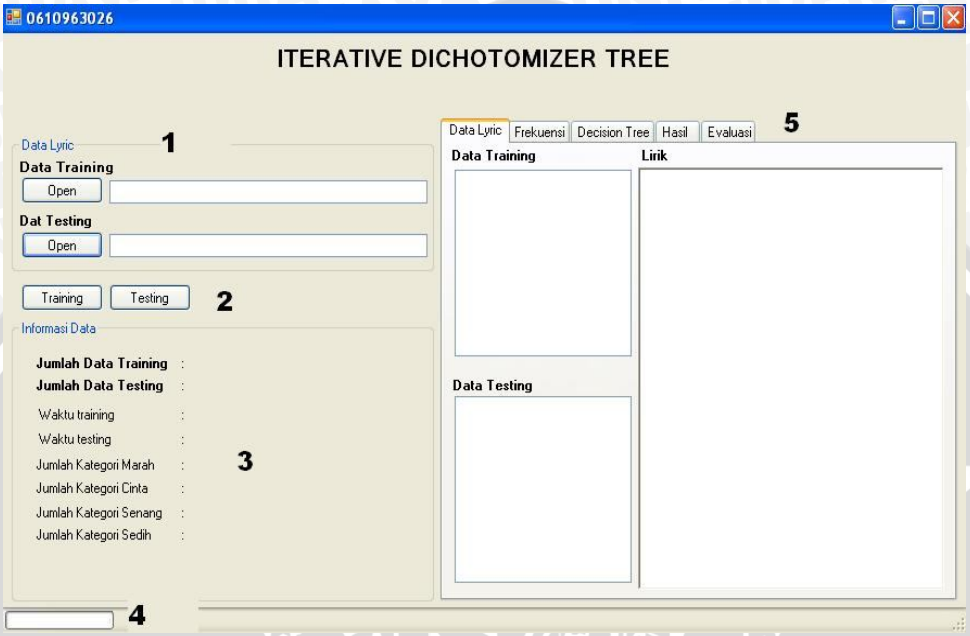
Pada data *testing* D10 atribut yang bisa masuk dalam aturan klasifikasi adalah atribut *love* dengan Int 1, aturan klasifikasi yang sesuai untuk kondisi D10 adalah aturan ke satu maka dokumen lirik lagu D10 berlabel *angry*.

Pada data *testing* D11 atribut yang bisa masuk dalam aturan klasifikasi adalah atribut *love* dengan Int 1, aturan klasifikasi yang sesuai untuk kondisi D11 adalah aturan ke satu maka dokumen lirik lagu D11 berlabel *angry*.

Pada data *testing* D12 atribut yang bisa masuk dalam aturan klasifikasi adalah atribut *love* dengan Int 1, aturan klasifikasi yang sesuai untuk kondisi D12 adalah aturan ke satu maka dokumen lirik lagu D12 berlabel *angry*.

3.5 Perancangan Antarmuka

Sistem dibuat dengan basis visual menggunakan bahasa pemrograman visual C#. Sistem memiliki satu *form* utama. Dari *form* utama tersebut terdapat beberapa bagian, berikut penjelasan bagian - bagian *form* dari gambar 3.18.



Gambar 3.18 Antarmuka sistem

Pada gambar 3.18 antarmuka sistem terdapat bagian dari *form* tersebut yg memiliki kegunaan berbeda, berikut keterangan dari bagian gambar 3.18.

1. *Button open* yang berfungsi untuk pengambilan data *testing* dan *training* dari sebuah *file* teks.
2. *Button* proses *training* dan *testing*, untuk menampilkan hasil klasifikasi yang dihasilkan sistem.
3. Bagian untuk menampilkan informasi data.
4. *Progress bar*, sebagai informasi untuk mengetahui lama proses berlangsung.
5. *Tab* yang berisikan data *lyric*, frekuensi, *decision tree*, hasil, evaluasi.

a. Data *lyric*, digunakan untuk menampilkan isi dari lirik lagu.
b. Frekuensi, digunakan untuk menampilkan hasil frekuensi kata pada lirik lagu.
c. *Decision tree*, digunakan untuk menampilkan pohon keputusan.
d. Hasil, digunakan menampilkan hasil dari pengklasifikasian lirik lagu tiap dokumen lirik lagu.
e. Evaluasi, untuk menampilkan jumlah dokumen yang cocok, dan besar akurasi.

3.6 Metode Pengujian

Setelah selesai melakukan pembuatan sistem, dilakukan pengujian terhadap sistem. Total keseluruhan data lirik lagu adalah 400 dokumen lirik lagu, dengan 100 dokumen lirik lagu pada kategori *angry, love, fun,* dan *sad*. Pada data *training* dan *testing* dikomposisikan jumlah yang sama antara data *angry, love, fun,* dan *sad*.

3.6.1 Perhitungan *Accuracy*

Dilakukan perhitungan *accuracy* dengan menggunakan persamaan 2.8. Berikut perancangan tabel penelitian untuk perhitungan *accuracy* terdapat pada tabel 3.12.

Tabel 3.12 Tabel perhitungan *accuracy*

Urutan Percobaan	Data Training	Data Testing	Akurasi
1			
…			
n			

BAB IV
IMPLEMENTASI

Pada bab ini dilakukan implementasi algoritma ID3 untuk mengklasifikasikan emosi berdasarkan lirik lagu dan analisis perangkat lunaknya.

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dijelaskan dalam sub bab ini adalah lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan perangkat lunak ini adalah sebagai berikut :

1. *Genuine Intel*® CPU U1400 @ 1,20GHz
2. Memori 1014MB RAM
3. *Harddisk* 75 GB
4. *Monitor* 11”
5. *Keyboard*
6. *Mouse*

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pembuatan sistem aplikasi pengelompokan emosi berdasarkan lirik lagu menggunakan algoritma *iterative dichotomizer tree* ini adalah *Microsoft Visual C#* sebagai *software development* dalam implementasi rancangan sistem.

4.2 Implementasi Program

Tampilan *form* utama untuk aplikasi sistem klasifikasi emosi berdasarkan lirik lagu dengan menggunakan metode ID3 dibagi menjadi 2 yakni bagian *training* dan *testing*. Secara garis besar proses dibagi menjadi 4 bagian utama implementasi, yaitu *preprocessing*, teks *transformation* dan *feature generation*, pembentukan pohon keputusan, dan keputusan ID3. Terdapat beberapa kelas yang

digunakan dalam program yakni kelas *DecisionTree*, *MainForm*, dan *Stemming*. Penjelasan kelas dapat dilihat pada tabel 4.1.

Tabel 4.1 Kelas pada program

Nama kelas	Penjelasan
DecisionTree.cs	Berisikan pembuatan *decision tree* dan proses ujinya.
MainForm.cs	Berisikan kelas utama dari program yang berisikan *method* penginputan data, *preprocessing*, dan data generalisasi serta pemanggilan kelas lain.
Stemming.cs	Berisikan *method* untuk memperoleh kata dasar pada sebuah kata.

4.2.1 Implementasi *Preprocessing*

Proses *preprocessing* ini dilakukan pada data *training* dan uji. Pada proses ini terdapat proses *tokenizing*, *stopword*, dan *stemming* ditambah dengan perhitungan frekuensi untuk proses selanjutnya yakni proses pembobotan. *Method* yang digunakan pada tahap *preprocessing* dapat dilihat pada tabel 4.2

Tabel 4.2 Method pada preprocessing

Method	Penjelasan
`private List<string> tokenizing(string isifile)`	Menghilangkan karakter yang tidak dibutuhkan, seperti tanda baca, angka, dan karakter spesial. - *Input*: *list string* kata - *Return*: *list string* kata
`Private List<string> stopword(List<string> token)`	Menghilangkan kata yang tidak penting atau tidak dianggap merepresentasikan isi dokumen. - *Input*: *list string* kata - *Return*: *list string* kata
`private List<string> stemming(List<string> term)`	Meperoleh kata dasar dari sebuah kata berimbuhan. - *Input*: *string* kata - *Return*: *string* kata
`private List<string> wordSingle(List<string> words)`	Mendaftar semua kata dan menghilangkan kata yang sama. - *Input*: *string* kata - *Return*: *string* kata
`private List<int> frekuensi(List<string> wordsNew, List<string> words)`	Menghitung frekuensi tiap kata. - *Input*: *list string* kata pada `wordSingle`, *list string* kata pada `stemming`. - *Return*: *list string* kata

4.2.1.1 *Tokenizing*

Tokenizing yang digunakan pada penelitian ini dibagi menjadi dua proses, yakni proses *case folding* dan *tokenizing*. *Case folding* adalah mengubah semua karakter huruf besar menjadi huruf kecil. Dalam proses ini karakter yang diterima adalah karakter huruf, sedangkan karakter lain seperti tanda baca akan diganti

dengan karakter spasi. *Sourcecode* untuk *case folding* dan *tokenizing* dapat dilihat pada *sourcecode* 4.1

```
private List<string> tokenizing(string isifile)
{isifile = isifile.ToLower();
            char[] pemisah = new char[] { '~', '}', '{', '|', '*', ':',
'\\', '_', '[', ']', '/', ';', '<', '>', ' ', '-', '=', '+', '.', ',',
', ', '"', '!', '?', '@', '#', '$', '%', '^', '&', '(', ')','\'', '\r',
'\n', '\t', '\v', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0' };
            List<string> token = isifile.Split(pemisah,
StringSplitOptions.RemoveEmptyEntries).ToList();
            return token;}
```

***Sourcecode* 4.1** *Method* proses *tokenizing*

Pada *method tokenizing* dilakukan *case folding* atau merubah menjadi huruf kecil, kemudian dibawa ke proses penghapusan karakter yang tidak dibutuhkan/tidak penting yang selanjutnya nilai dikembalikan pada variabel *token*.

4.2.1.2 *Stopwords*

Tujuan proses ini adalah untuk memperoleh kata-kata penting dengan cara membuang kata-kata yang dianggap tidak penting. Proses *stopword* dapat dilihat pada *sourcecode* 4.2.

```
private List<string> stopword(List<string> token)
        {
            string[] stopword = new string[1000];
            int m = stopword.GetLength(0);
            int n = token.Count();
            string[] _ceksplit = new string[n];
            bool sama = true;
            int jmlh = 0;
            FileStream sr = new FileStream("stopwords.txt",
FileMode.Open);
            StreamReader str = new StreamReader(sr);
            int a = 0;
            while (!str.EndOfStream)
            {
                stopword[a] = str.ReadLine();
                a++;
            }
            sr.Close();
            str.Close();
            for (int j = 0; j < n; j++)
            {
                sama = true;
                for (int k = 0; k < m; k++)
                {
                    if (token[j] == stopword[k])
                        sama = false;
                }
                if (sama != false)
                {
                    _ceksplit[j] = token[j];
                    jmlh++;
                }
            }
            List<string> words = new List<string>();
```

```
            for (int l = 0; l < n; l++)
            {
                if (_ceksplit[l] != null)
                {
                    words.Add(_ceksplit[l]);
                }
            }
            return words;}
```

***Sourcecode* 4.2** *Method* proses *stopword*

Pada proses ini daftar *stopword* sendiri disimpan di dalam file "*stopwords.txt*" yang kemudian diambil untuk pengecekan kata apakah *token* atau kata hasil *tokenizing* merupakan *stopword*, jika tidak termasuk *stopword* maka kata tersebut akan ditampung ke *words*.

4.2.1.3 *Stemming*

Pada proses *stemming* memanfaatkan kelas dari *porter stemmer* untuk mendapatkan kata dasar dari suatu kata. Kelas *porter stemmer* diambil dari *http://www.tartarus.org/~martin/PorterStemmer.* Pemanggilan kelas porter *stemmer* dapat dilihat pada *sourcecode* 4.3.

```
private List<string> stemming(List<string> term)
{
    List<string> stem_term = new List<string>();
    Stemmer stem = new Stemmer();
    for (int i = 0; i < term.Count; i++)
    {stem_term.Add(stem.Porter.stemTerm(term[i]));}
    return stem_term;}
```

***Sourcecode* 4.3** *Porter Stemmer*

4.2.1.4 Frekuensi Kata

Perhitungan frekuensi kata dilakukan setelah kata atau *term* berbentuk kata dasar. Pada perhitungan frekuensi kata terdapat dua subproses, yakni proses pembentukan *list* kata untuk tiap dokumen lirik lagu dan proses penghitungan jumlah frekuensi dari kata tersebut. Pembentukan kata dapat dilihat pada *sourcecode* 4.4.

```
private List<string> wordSingle(List<string> words)
        {
            var sList = new ArrayList();
            for (int i = 0; i < words.Count; i++)
            {
                if (sList.Contains(words[i]) == false){
                    sList.Add(words[i]);}}
            var wordsNew_ = sList.ToArray();
            string[] wordsNew = new string[wordsNew_.Length];
            for (int i = 0; i < wordsNew_.Length; i++){
                wordsNew[i] = (string)wordsNew_[i]; }
            return wordsNew.ToList();}
```

***Sourcecode* 4.4** *Method list* kata tiap dokumen

Pada *method* pembentukan *list* kata dilakukan pembuangan duplikasi kata yang selanjutnya digunakan untuk proses perhitungan frekuensi kata. *Method* perhitungan frekuensi kata dapat dilihat pada *sourcecode* 4.5.

```
private List<int> frekuensi(List<string> wordsNew, List<string> words)
        {
            int[] wordsFrek = new int[wordsNew.Count];
            for (int i = 0; i < wordsNew.Count; i++)
            {
                for (int j = 0; j < words.Count; j++)
                {
                    if (words[j].Equals((string)wordsNew[i]))
                    {
                        wordsFrek[i]++;
                    }
                }
            }
            return wordsFrek.ToList();
        }
```

***Sourcecode* 4.5** *Method* perhitungan frekuensi kata

Perhitungan frekuensi dilakukan dengan menghitung kata pada `wordsNew` yang terdapat di list `words`, selanjutnya jumlah frekuensi disimpan ke dalam `wordsFrek`.

4.2.2 Implementasi Pembobotan

Sebelum proses pembobotan dilakukan, kata akan disaring lagi untuk kata yang mempunyai frekuensi lebih besar dari 2. Pada proses ini akan didapatkan kata baru yang nantinya akan dijadikan atribut untuk *decision tree*. Proses penyaringan kata dapat dilihat pada *sourcecode* 4.6

```
for (int j = 0; j < pro_wsgl.Count; j++)
                {
                    if (pro_frek[j] > 2)
                    {
                        for (int k = 0; k < nTerm; k++)
                        {
                            if (!term.Contains(pro_wsgl[j])){
                                term.Add(pro_wsgl[j]);
                                dataTraining[i, nTerm] = pro_frek[j];
                                nTerm++;
                                break;}
                            if (term[k].Equals(pro_wsgl[j])){
                                dataTraining[i, k] = pro_frek[j];}}
                        if (nTerm == 0){
                            term.Add(pro_wsgl[j]);
                            dataTraining[i, nTerm] = pro_frek[j];
                            nTerm++;
                        }}}}
```

***Sourcecode* 4.6** *Filtering* kata

Term dari hasil frekuensi tiap data akan diseleksi untuk frekuensi lebih besar dari 2, setelah itu kata tersebut disimpan didalam `term`. Kemudian *term* beserta jumlah frekuensi akan disimpan ke dalam `dataTraining`, selanjutnya dipergunakan untuk perhitungan pembobotan. Perhitungan bobot dapat dilihat pada *sourcecode* 4.7.

```
int df = 0;
            double[,] dataPembobotan = new double[lb_teks.Items.Count,
nTerm];
            for (int i = 0; i < nTerm; i++)
            {
                for (int j = 0; j < lb_teks.Items.Count; j++)
                {
                    if (dataTraining[j, i] != 0)
                    {
                        df++;
                    }
                }
                dataTraining[lb_teks.Items.Count, i] = df;
                df = 0;
                dataTraining[(lb_teks.Items.Count + 1), i] =
Math.Round(Math.Log10(lb_teks.Items.Count /
dataTraining[lb_teks.Items.Count, i]), 4,
MidpointRounding.AwayFromZero);
                idf.Add(dataTraining[(lb_teks.Items.Count + 1), i]);
                for (int k = 0; k < lb_teks.Items.Count; k++){
                    dataPembobotan[k, i] = dataTraining[k, i] *
dataTraining[(lb_teks.Items.Count + 1), i];
                }}
```

***Sourcecode* 4.7** Proses pembobotan

Pada proses perhitungan bobot dicari terlebih dahulu nilai *df*. Nilai *df* diambil berdasarkan jumlah ada atau tidaknya sebuah *term* disetiap dokumen, kemudian nilai *df* disimpan kedalam `dataTraining` pada *index* setelah dokumen. Setelah *df* diketahui, kemudian dihitung nilai *idf* dan disimpan kedalam `dataTraining` pada *index* setelah *df*. Data pembobotan diperoleh dengan mengalikan *tf* dan *idf* kemudian disimpan kedalam `dataPembobotan`.

4.2.3 Implementasi Transformasi Data

Transformasi data bertujuan untuk memberi kelas interval dari nilai hasil pembobotan. Terdapat dua proses pada transformasi data yakni proses ekstraksi data dan distribusi frekuensi. Proses ekstraksi data dapat dilihat pada *sourcecode* 4.8.

```
toolStripProgressBar1.Value += 20;
            double max = dataPembobotan.Cast<double>().Max();
            double min = dataPembobotan.Cast<double>().Min();
            double range = max - min;
            kelas = 1 + (3.3 *
Math.Log10(dataPembobotan.Cast<double>().Count()));
            kelas = Math.Ceiling(kelas);
            double pInterval = Math.Round(range / kelas, 4,
MidpointRounding.AwayFromZero);
            inv _interval = new inv();
            for (int i = 0; i <= kelas; i++)
            {
                if (i == 0)
                {
                    _interval.a = min;
                    _interval.b = _interval.a + pInterval;
                    interval.Add(_interval);
                }
                else
                {
                    _interval.a = interval[i - 1].b + 0.0001;
                    _interval.b = _interval.a + pInterval;
                    interval.Add(_interval);
                }
            }
```

***Sourcecode* 4.8** *Ekstraksi* data

Pada proses ekstraksi data dicari nilai *range* dari bobot, kelas dan panjang interval. Setelah diperoleh nilai *range*, selanjutnya akan dimasukkan ke dalam kelas interval sejumlah kelas ke dalam variabel `interval`. Variabel `interval` digunakan untuk memberikan label interval pada proses distribusi frekuensi. Proses distribusi frekuensi dapat dilihat pada *sourcecode* 4.9.

```
string[,] dataKategorisasi = new string[lb_teks.Items.Count, nTerm];
            var hasil = new List<bool>();
            for (int i = 0; i < lb_teks.Items.Count; i++)
            {
                for (int j = 0; j <= nTerm; j++)
                {
                    if (j == nTerm)
                    {
                        if (lb_teks.Items[i].ToString().Contains("ang"))
  {                       sb.Append("angry");
                        }
                        if (lb_teks.Items[i].ToString().Contains("lov"))
                        {
                            sb.Append("love");
                        }
                        if (lb_teks.Items[i].ToString().Contains("fun"))
                        {
                            sb.Append("fun");
                        }
                        if (lb_teks.Items[i].ToString().Contains("sad"))
                        {
                            sb.Append("sad");
                        }}
                    else
                    {
                        for (int k = 0; k <= kelas; k++)
```

```
                    {
                        if ((dataPembobotan[i, j] >= interval[k].a)
&& (dataPembobotan[i, j] <= interval[k].b) && (k != kelas))
                        { dataKategorisasi[i, j] = (string)("int" +
(k + 1)); }
                        if ((dataPembobotan[i, j] >= interval[k].a)
&& (k == kelas))
                        { dataKategorisasi[i, j] = (string)("int" +
(k + 1)); }
                    }
                    sb.Append(dataKategorisasi[i, j] + ",");}}
if (i < (lb_teks.Items.Count - 1))    sb.Append("\n");}
```

***Sourcecode* 4.9** Distribusi frekuensi

Pada proses distribusi frekuensi dilakukan pengecekan pada bobot pada setiap kata dan setiap dokumen lirik lagu, misalkan jika bobot tersebut diantara nilai interval ke-1 maka akan diberi label “int1” dan seterusnya. Data hasil proses kelas interval disimpan ke dalam `dataKategorisasi`.

4.2.4 Implementasi Pembentukan Pohon Keputusan

Pada proses pembentukan pohon keputusan, didalam program digunakan beberapa *method* untuk membuat pohon keputusan. *Method* yang digunakan untuk membentuk pohon keputusan terdapat pada tabel 4.3

Tabel 4.3 *Method* pembentukan pohon keputusan

`private double getCalculatedEntropy(int angry, int love, int fun, int sad)`	Menghitung *entropy* kelas. - *Input*: int *angry*, int *love*, int *fun*, int *sad* - *Return*: double *result*
`private double gain(DataTable samples, TreeAttribute attribute)`	Menghitung gain. - *Input*: DataTable *sample* data, TreeAttribute atribut tree - *Return*: double *result*
`private TreeNode internalMountTree(DataTable samples, string targetAttribute, TreeAttributeCollection attributes)`	Membentuk pohon keputusan - *Input*: DataTable *sample* data, string atribut tujuan, TreeAttributeCollection kumpulan atribut. - *Return*: Tree*Node* tree.

Pembentukan pohon keputusan pada metode ID3 dilakukan dengan menghitung nilai *gain* pada setiap atribut yang nantinya akan diambil nilai terbesar untuk membentuk pohon keputusan. Untuk menghitung *gain*, terlebih dahulu harus menghitung *entropy* atributnya. Proses perhitungan *entropy* dapat dilihat pada *sourcecode* 4.10.

```
private double getCalculatedEntropy(int angry, int love, int fun, int
sad)
        {
            int total = angry + love + fun + sad;
            double ratioAngry = (double)angry / total;
            double ratioLove = (double)love / total;
            double ratioFun = (double)fun / total;
            double ratioSad = (double)sad / total;
            if (ratioAngry != 0)
                ratioAngry = -(ratioAngry) * System.Math.Log(ratioAngry,
2);
            if (ratioLove != 0)
                ratioLove = -(ratioLove) * System.Math.Log(ratioLove, 2);
            if (ratioFun != 0)
                ratioFun = -(ratioFun) * System.Math.Log(ratioFun, 2);
            if (ratioSad != 0)
                ratioSad = -(ratioSad) * System.Math.Log(ratioSad, 2);

            double result = ratioAngry + ratioLove + ratioFun + ratioSad;
            return result;
        }
```

Sourcecode 4.10 Perhitungan *entropy*

Pada perhitungan *entropy* data dibagi 4, yakni *angry*, *sad*, *love*, *fun*. Dilakukan perhitungan rasio tiap kategori, kemudian dikalkulasi untuk memperoleh nilai *entropy* kelasnya. *Entropy* hasil perhitungan tersebut akan digunakan untuk menghitung *information gain*. Proses perhitungan *information gain* dapat dilihat pada *sourcecode* 4.11.

```
private double gain(DataTable samples, TreeAttribute attribute)
        {
            PossibleValueCollection values = attribute.PossibleValues;
            double sum = 0.0;

            for (int i = 0; i < values.Count; i++)
            {
                int angry, love, fun, sad;

                angry = love = fun = sad = 0;

                getValuesToAttribute(samples, attribute, values[i], out
angry, out love, out fun, out sad);

                double entropy = getCalculatedEntropy(angry, love, fun,
sad);
                sum += -(double)(angry + love + fun + sad ) / mTotal *
entropy;
            }
            return mEntropySet + sum;
        }
```

***Sourcecode* 4.11** Perhitungan *information gain*

Perhitungan *gain* dilakukan dengan mengambil data *love*, *fun*, *sad*, dan *angry* dengan *method* `getValuesToAttribute`, kemudian nilai tersebut digunakan untuk memanggil *method* `getCalculatedEntropy` untuk mendapatkan nilai *entropy* dan digunakan untuk menghitung nilai *gain*. Nilai *gain* terbesar akan dijadikan *root* atau atribut pada pohon keputusan. Proses pembuatan pohon keputusan dapat dilihat pada *sourcecode* 4.12.

```
private TreeNode internalMountTree(DataTable samples, string
targetAttribute, TreeAttributeCollection attributes){
            if (samplesAreAngry(samples, targetAttribute) == true)
                return new TreeNode(new OutcomeTreeAttribute("angry"));
            if (samplesAreLove(samples, targetAttribute) == true)
                return new TreeNode(new OutcomeTreeAttribute("love"));
            if (samplesAreFun(samples, targetAttribute) == true)
                return new TreeNode(new OutcomeTreeAttribute("fun"));
            if (samplesAreSad(samples, targetAttribute) == true)
                return new TreeNode(new OutcomeTreeAttribute("sad"));
            if (attributes.Count == 0)
                return new TreeNode(new
OutcomeTreeAttribute(getMostCommonValue(samples, targetAttribute)));
            mTotal = samples.Rows.Count; // menhitung jumlah data
            mTargetAttribute = targetAttribute; // target klas atribut
            mTotalAngry = countTotalAngry(samples);
            mTotalLove = countTotalLove(samples);
            mTotalFun = countTotalFun(samples);
            mTotalSad = countTotalSad(samples);
            mEntropySet = getCalculatedEntropy(mTotalAngry, mTotalLove,
mTotalFun, mTotalSad);
            TreeAttribute bestAttribute = getBestAttribute(samples,
attributes);
            TreeNode root = new TreeNode(bestAttribute);
            if (bestAttribute == null)
                return root;
            DataTable aSample = samples.Clone();
            foreach (string value in bestAttribute.PossibleValues){
                aSample.Rows.Clear();
                DataRow[] rows =
samples.Select(bestAttribute.AttributeName + " = " + "'" + value + "'");
                foreach (DataRow row in rows){
                    aSample.Rows.Add(row.ItemArray);}
                TreeAttributeCollection aAttributes = new
TreeAttributeCollection();
                for (int i = 0; i < attributes.Count; i++){
                    if (attributes[i].AttributeName !=
bestAttribute.AttributeName)
                        aAttributes.Add(attributes[i]);}
                if (aSample.Rows.Count == 0){
                    return new TreeNode(new
OutcomeTreeAttribute(getMostCommonValue(aSample, targetAttribute)));}
                else{
                    DecisionTree dc3 = new DecisionTree();
                    TreeNode ChildNode = dc3.mountTree(aSample,
targetAttribute, aAttributes);
                    root.AddTreeNode(ChildNode, value);}}
            return root; }
```

***Sourcecode* 4.12** Pembuatan pohon keputusan

Pada pembuatan pohon keputusan, pertama dokumen akan dicek apakah semua *angry*, *love, fun,* atau *sad*. Jika data semua *angry* maka pohon sudah mencapai ujung dan diberi *value "angry"*, jika data semua *love* maka pohon sudah mencapai ujung dan diberi *value "love"*, dan seterusnya hingga iterasi akan dihentikan untuk cabang yang bersangkutan. Apabila dokumen terdapat data *angry, love, fun,* maupun *sad,* maka dilakukan perhitungan *gain* untuk setiap atributnya dan diambil nilai terbaik atau terbesar dengan memanggil *method* `getBestAttribute` untuk dijadikan ujung berikutnya.

4.2.5 Implementasi Keputusan ID3

Pohon keputusan yang telah terbuat, selanjutnya akan digunakan untuk memutuskan apakah lirik lagu tersebut merupakan emosi *angry, love, fun,* dan *sad*. Proses keputusan ID3 dapat dilihat pada *sourcecode* 4.13.

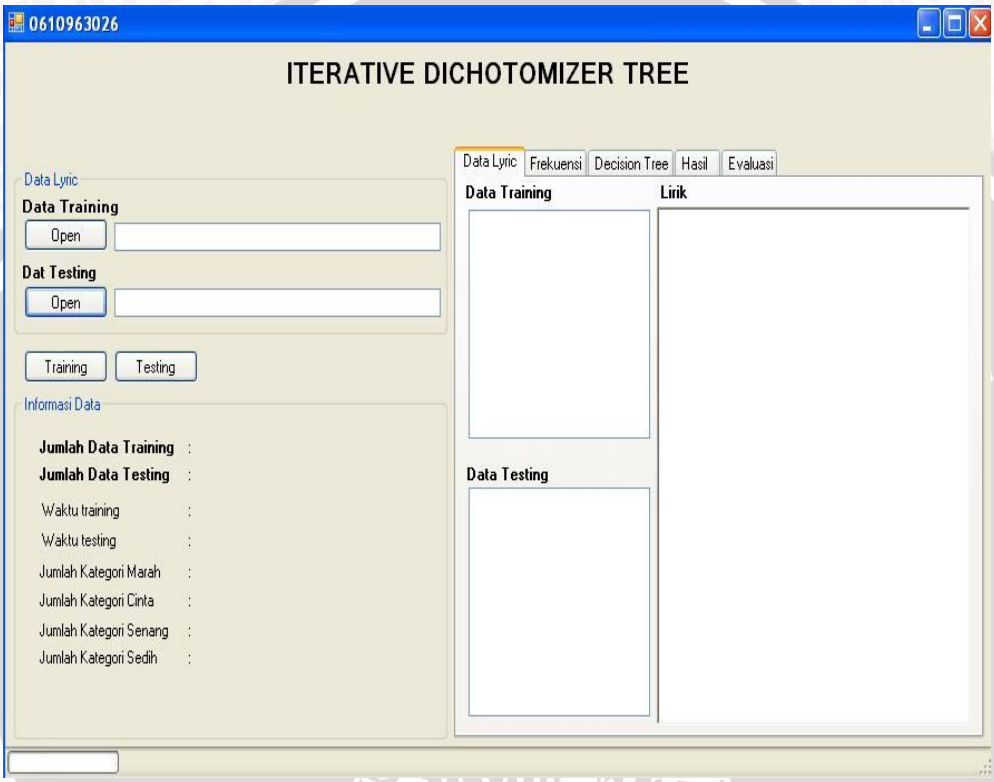
```
public string PrintResult(TreeNode root, List<string>
dataInv, List<string> term){
            if (root.Attribute.PossibleValues == null){
                result += root.Attribute;}
            else{for (int i = 0; i < term.Count; i++){if
(term[i].Equals(root.Attribute.AttributeName)){
                        if (root.Attribute.PossibleValues !=
null){for (int j = 0; j <
root.Attribute.PossibleValues.Count; j++){
                                if
(dataInv[i].Equals(root.Attribute.PossibleValues[j])){
                                    TreeNode childNode =
root.GetChildByBranchName(root.Attribute.PossibleValues[j]);
                                    if (childNode.Attribute
!= null){result = PrintResult(childNode, dataInv, term);}}}
                            if (result == String.Empty)
                                return "result";}} }}return
result;}
```

***Sourcecode* 4.13** Keputusan ID3

Pada proses keputusan ID3 dicek terlebih dahulu apakah atribut mempunyai cabang lagi atau tidak, jika tidak maka atribut tersebut adalah ujung dari *tree* yang berisi "*angry*", "*love*", "*fun*", atau "*sad*". Jika atribut mempunyai cabang, maka akan dicek *term* termasuk interval berapa. Apabila didalam atribut *value* tidak terdapat nilai interval yang sama dengan interval *term testing*, maka nilai `result` adalah "*angry*" dan seterusnya. Nilai interval pada *term testing* yang terdapat pada atibut *value* akan diproses ke atribut selanjutnya sampai atribut mempunyai *value* kosong.

4.3 Penerapan Aplikasi

Berdasarkan perancangan antarmuka pada subbab 3.3. Pada tahap *training* dan *testing*, data yang telah diambil akan ditampilkan pada *tab* data *lyric* dan frekuensi. Pada gambar 4.1 *tab* data *lyric* ditampilkan isi teks yang dipilih pada *list*, sedangkan *tab* frekuensi menampilkan hasil frekuensi untuk dokumen yang dipilih pada *list*.



Gambar 4.1 Antarmuka Sistem

Pada gambar 4.1, *tab* hasil, dan evaluasi berfungsi setelah tombol *testing* dijalankan. *Tab decision tree* untuk menampilkan *decision tree* hasil *training*. Jumlah data *training*, *angry*, *love*, *fun*, dan *sad* akan menampilkan hasilnya di informasi data ketika pengguna mengambil data *training*.

Pada proses *testing* berupa hasil klasifikasi dan evaluasi. *Tab* hasil akan menampilkan hasilnya ketika tombol proses dijalankan, kemudian jumlah dokumen yang cocok dan akurasi akan dihitung dan ditampilkan pada bagian *tab* evaluasi.

BAB V
ANALISA DAN PEMBAHASAN

5.1 Skenario Pengujian

Pada sistem pengklasifikasian emosi berdasarkan lirik lagu berbahasa inggris dengan menggunakan metode ID3 akan dilakukan pengujian. Pengujian dilakukan dengan menggunakan beberapa data *testing* dengan kondisi parameter tertentu, untuk mengetahui kehandalan yang dihasilkan oleh aplikasi.

5.1.1 Data pengujian

Data yang digunakan dalam pengujian berformat teks atau berupa teks.*txt*. Total keseluruhan data lirik lagu adalah 400 dokumen lirik lagu, dengan 100 dokumen lirik lagu pada kategori *angry*, *fun*, *love*, dan *sad*. Data yang digunakan berjumlah 20, 40, 80, 120, 160, dan 200 dokumen lirik lagu. Pada tabel 5.1, data *testing* berjumlah 20 dokumen lirik lagu, pengujian tetap menggunakan data yang sama, sedangkan pada tabel 5.2 data training berjumlah 20 dokumen lirik lagu menggunakan data yang sama, untuk mendapatkan keterkaitan antar pengujian. Pada tabel 5.3 *training* dan *testing* dengan perbandingan sama.

5.1.2 Lingkungan Pengujian

Jumlah data sebanyak 20, 40, 80, 120, 160, dan 200 dokumen lirik lagu, dengan komposisi perbandingan data *angry*, *fun*, *love*, dan *sad* sebesar 50:50 untuk data *training* maupun *testing*. Pengujian data hanya dilakukan 200 dokumen lirik lagu data *training*. Pada saat dilakukan pengujian data *training* dan *testing* sama dengan data > 150, pengujian data tidak dapat dilakukan karena keterbatasan perangkat keras.

5.1.3 Hasil Pengujian

Hasil pengujian aplikasi pengklasifikasian emosi berdasarkan lirik lagu berbahasa Inggris dengan menggunakan metode ID3 didapatkan dari hasil skenario pengujian pada subbab 3.6.1. Setiap pengujian didapatkan hasil kecocokan perhitungan sistem dengan perhitungan manual, selanjutnya dihitung

akurasi dengan menghitung prosentase jumlah data yang benar terhadap keseluruhan data. Hasil perhitungan klasifikasi dapat dilihat pada tabel 5.1, 5.2. dan 5.3.

Tabel 5.1 Hasil klasifikasi *training* lebih besar dari *testing*

Data Training	Data Testing	Uji 1	Uji 2	Uji 3	Rata-rata
40	20	40%	20%	15%	25%
80	20	60%	40%	40%	47%
120	20	70%	65%	55%	63%
160	20	65%	60%	60%	62%
200	20	60%	45%	55%	53%

Berdasarkan tabel 5.1, pengujian dilakukan 3 kali uji coba. Tiap pengujian menggunakan jumlah data *training* dan *testing* yang sama, tetapi berbeda dokumen lirik lagu. Pengaruh jumlah data latih dan data *testing* terhadap hasil akurasi didapatkan jumlah data latih dan data *testing* terbaik pada pengujian ini ketika data latih berjumlah 120 dan data *testing* 20 dari data keseluruhan dengan rata-rata akurasi 63%.

Tabel 5.2 Hasil klasifikasi *testing* lebih besar dari *training*

Data Training	Data Testing	Uji 1	Uji 2	Uji 3	Rata-rata
20	40	22.50%	20%	17.50%	20%
20	80	20%	26.20%	28.70%	25%
20	120	24.10%	21.60%	22.50%	23%
20	160	21.20%	19.30%	21.90%	21%
20	200	23.50%	22.50%	21.50%	23%

Berdasarkan tabel 5.2, pengujian dilakukan 3 kali uji coba. Tiap pengujian menggunakan jumlah data *training* dan data *testing* yang sama, tetapi berbeda dokumen lirik lagu. Pengaruh jumlah data latih dan data *testing* terhadap hasil akurasi didapatkan jumlah data latih dan data *testing* terbaik pada pengujian ini ketika data latih berjumlah 20 dan data *testing* 80 dari data keseluruhan dengan rata-rata akurasi 25%.

Tabel 5.3 Hasil klasifikasi *training* dan *testing* dengan perbandingan sama

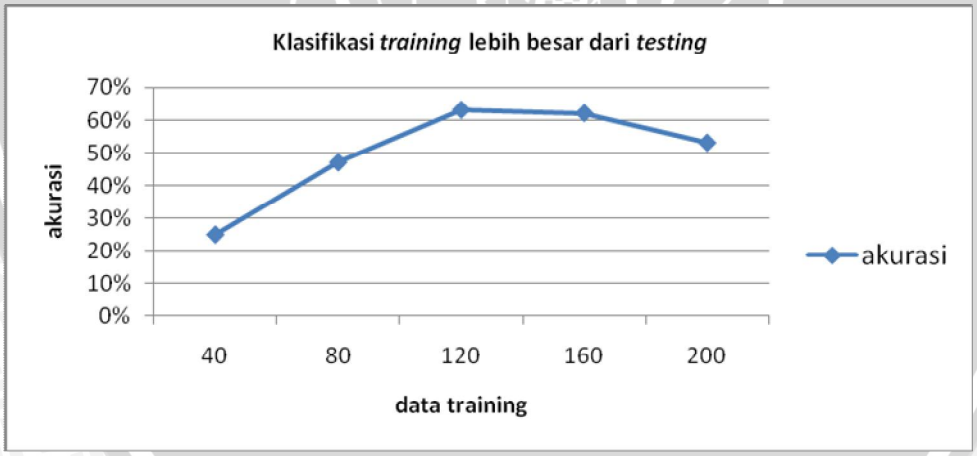
Data Training	Data Testing	Uji 1	Uji 2	Uji 3	Rata-rata
20	20	35%	30%	35%	33%
40	40	53%	55%	48%	52%
80	80	64%	63%	61%	62%
120	120	58%	58%	54%	57%

Berdasarkan tabel 5.3, pengujian dilakukan 3 kali uji coba. Tiap pengujian menggunakan jumlah data *training* dan data *testing* yang sama, tetapi berbeda dokumen lirik lagu. Pengaruh jumlah data latih dan data *testing* terhadap hasil akurasi didapatkan jumlah data latih dan data *testing* terbaik pada pengujian ini ketika data latih berjumlah 80 dan data *testing* 80 dari data keseluruhan dengan rata-rata akurasi 62%.

5.2 Analisa Hasil

Dari data hasil pengujian pada tabel 5.1, 5.2, 5.3, dapat dibuat grafik sebagai analisa data.

5.2.1 Analisa hasil klasifikasi *training* lebih besar dari *testing*

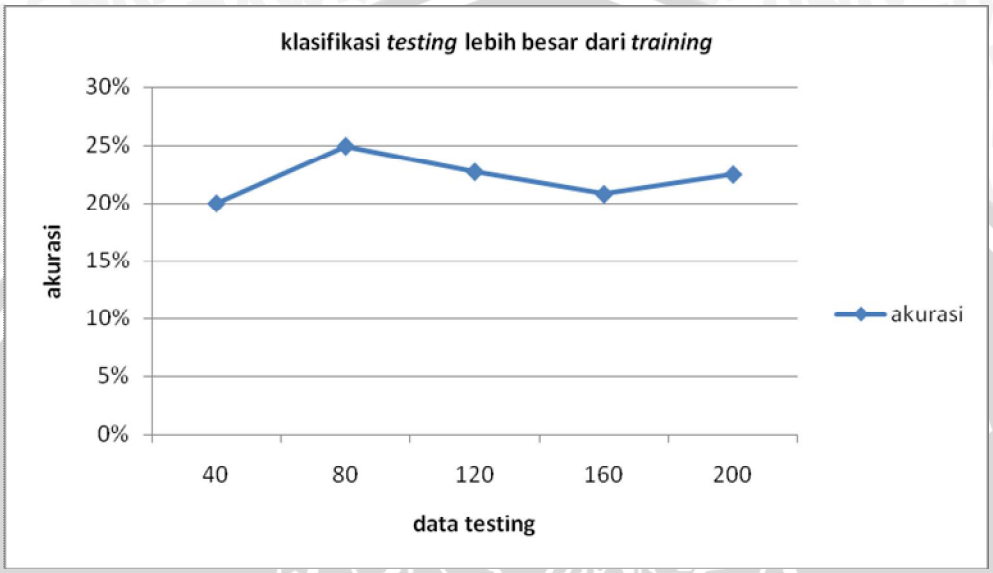


Gambar 5.1 Grafik hasil akurasi klasifikasi *training* lebih besar dari *testing*

Pada gambar 5.1 hasil akurasi, dapat dilihat hasil tertinggi dari rata-rata akurasi pada data *training* 120 dan data *testing* 20 sebesar 63% karena pembelajaran data *training* lebih optimal, sedangkan pada data *training* 40 dan data *testing* 20 lebih rendah sebesar 25% karena kurangnya pembelajaran pada

data latih. Dan pada data *training* 160 *testing* 20 mengalami sedikit penurunan hasil akurasi sebesar 62%, karena perbandingan jumlah data latih dan data *testing* terlalu jauh.

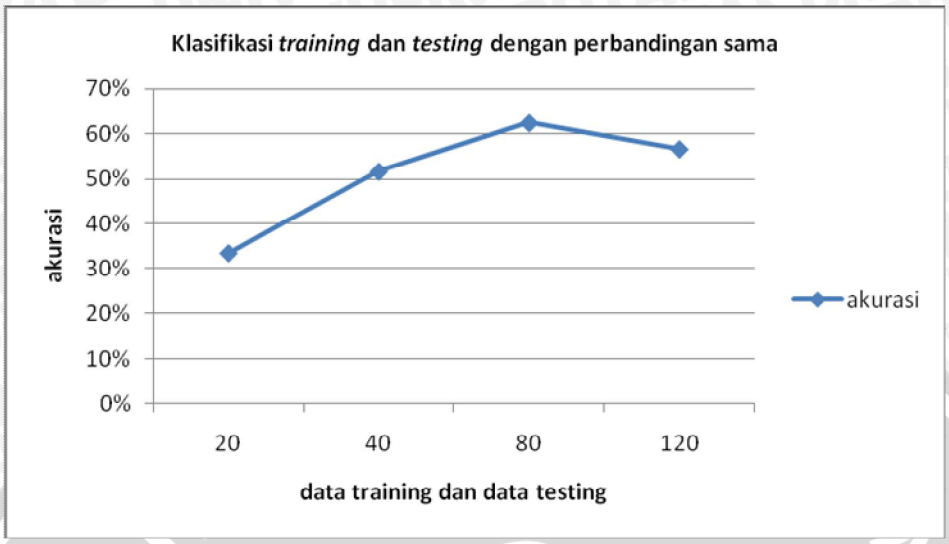
5.2.2 Analisa hasil klasifikasi *testing* lebih besar dari *training*



Gambar 5.2 Grafik hasil akurasi klasifikasi *testing* lebih besar dari *training*

Pada gambar 5.2 hasil akurasi, dapat dilihat hasil akurasi tertinggi dari rata-rata akurasi pada data *training* 20 dan data *testing* 80 sebesar 25%, sedangkan pada data *training* 20 dan data *testing* 40, dan pada data *training* 20 dengan data *testing* 160 lebih rendah dengan masing-masing sebesar 20%, dan 21%. Dan pada data *training* 20 *testing* 120, serta data *training* 20 *testing* 200 hasil akurasi sama sebesar 23%. Dapat dilihat hasil akurasi tidak lebih dari 25% karena tidak mempunyai cukup pengetahuan terhadap suatu data *testing* dan data *training* kurang banyak.

5.2.3 Analisa hasil klasifikasi *training* dan *testing* dengan perbandingan sama



Gambar 5.3 Grafik hasil akurasi klasifikasi *training* dan *testing* dengan perbandingan sama

Berdasarkan gambar 5.3 didapatkan bahwa hasil dari pengujian dengan jumlah data latih dan *testing* sama, dapat dilihat hasil tertinggi dari rata-rata akurasi pada data dengan jumlah 80 sebesar 62% karena pembelajaran data *training* lebih optimal, sedangkan pada data dengan jumlah 20 lebih rendah sebesar 33% karena kurangnya pembelajaran pada data latih. Dan dengan jumlah data 120 mengalami penurunan hasil akurasi sebesar 57%, karena adanya kemiripan kategori pada atribut lirik lagu, dimana ketingkatan akurasi yang dihasilkan ada ketergantungan data dari setiap dokumen.

Analisis hasil percobaan yang telah dilakukan, dapat dilihat pada tabel 5.1 hasil akurasi tertinggi dari rata-rata akurasi pada data *training* 120 dan data *testing* 20 sebesar 63% karena pembelajaran data *training* lebih optimal, sedangkan pada data *training* 40 dan data *testing* 20, dan pada data *training* 80 dengan data *testing* 20 lebih rendah dengan masing-masing sebesar 25%, dan 47% karena kurangnya pembelajaran pada data latih. Dan pada data *training* 160 *testing* 20, serta data *training* 200 *testing* 20 mengalami sedikit penurunan dengan hasil akurasi masing-masing sebesar 62% dan 53%, karena perbandingan jumlah data latih dan data *testing* terlalu jauh.

Sedangkan pada tabel 5.2, dapat dilihat hasil akurasi tertinggi dari rata-rata akurasi pada data *training* 20 dan data *testing* 80 sebesar 25%, sedangkan pada data *training* 20 dan data *testing* 40, dan pada data *training* 20 dengan data *testing* 160 lebih rendah dengan masing-masing sebesar 20%, dan 21%. Dan pada data *training* 20 *testing* 120, serta data *training* 20 *testing* 200 hasil akurasi sama sebesar 23%. Dapat dilihat hasil akurasi tidak lebih dari 25% karena tidak mempunyai cukup pengetahuan terhadap suatu data *testing* dan data *training* kurang banyak.

Pada tabel 5.3, didapatkan bahwa hasil dari pengujian dengan jumlah data latih dan *testing* sama, dapat dilihat hasil tertinggi dari rata-rata akurasi pada data dengan jumlah 80 sebesar 62% karena pembelajaran data *training* lebih optimal, sedangkan pada data dengan jumlah 20 dan 40 lebih rendah dengan masing-masing sebesar 33% dan 52% karena kurangnya pembelajaran pada data latih. Dan dengan jumlah data 120 mengalami penurunan hasil akurasi sebesar 57%, karena adanya kemiripan kategori pada atribut lirik lagu, dimana ketingkatan akurasi yang dihasilkan ada ketergantungan data dari setiap dokumen.

Berdasarkan eksperimen yang telah dilakukan, pengujian dengan menggunakan data *training* lebih besar dari *testing* cukup sesuai untuk melakukan klasifikasi emosi lirik berdasarkan lirik lagu. Hal ini terlihat dari nilai rata-rata tingkat akurasi yang didapatkan sebesar 63%.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Dari hasil perancangan dan implementasi sistem maka dapat disimpulkan:

1. Telah diimplementasikan metode pengklasifikasian dokumen teks berbahasa inggris dapat diaplikasikan dengan menggunakan metode *iterative dichotomizer tree.* ID3 memerlukan proses *preprocessing,* pembobotan dan generalisasi data sebelum melakukan proses pengenalan pola.
2. Tingkat rata-rata akurasi terbaik dari pengklasifikasian emosi lagu menggunakan metode ID3 adalah sebesar 63% pada pengujian *training* lebih besar dari *testing,* sedangkan pada pengujian *testing* lebih besar dari *training* sebesar 25%. Serta pada pengujian *training* dan *testing* dengan jumlah sama sebesar 62%. Semakin besar jumlah data *training* dibandingkan dengan data *testing,* maka hasil akurasi yang dihasilkan semakin besar. Dan ketika jumlah data *testing* lebih besar dari data *training,* hasil akurasi yang dihasilkan kurang optimal. Hal ini dikarenakan adanya variasi pola data latih dengan data uji yang berbeda jauh sehingga data training yang diambil tidak dapat mewakili pola data testing.

6.2 Saran

Dalam penelitian ini dapat dikembangkan untuk penelitian lebih lanjut, diantaranya:

Beberapa kata yang mengekspresikan emosi mempunyai kesamaan dengan kata lain. Untuk itu perlu juga dilakukan pengecekan terhadap makna lain dari sebuah kata agar lirik lagu yang mempunyai kata berbeda dengan makna sama dapat mempunyai nilai similaritas yang cukup tinggi.

Daftar Pustaka

1. [AKB-08] Akbar, Ali. 2008, "*System Automatic Music Emotion Classification*", Program Studi Informatika, Institut Teknologi Bandung, Bandung.
2. [ARI-11] Ariastuti, L.,Siwi, 2011, "Pengelompokkan *Mood* Lagu Berdasarkan Lirik Lagu Menggunakan Algoritma Semut", Program Studi Ilmu Komputer, Universitas Brawijaya, Malang.
3. [BRE-84] Breiman, L., Friedman, J., H., Olshen, R., A., & Stone, P., J., 1984, "*Classification and Regression tree*", Belmont, CA: Wadsworth International Group.
4. [DEF-10] Defiyanti, Sofi dan Pardede, Crispina D., L., 2010, "Perbandingan Kinerja Algoritma ID3 dan C4.5 Dalam Klasifikasi *Spam-Mail*", Universitas Gunadarma, Depok.
5. [ELD-08] Eldira, Hervilorra, 2008, "*Web Mining* Untuk Pencarian Dokumen Bahasa Inggris Menggunakan *Hill Climbing Automatic Cluster*", Institut Teknologi Sepuluh November, Surabaya.
6. [EVE-02] Even,Yahir dan Zohar, 2002, "*Introduction to Teks Mining. Automated Learning Group National Center For Supercomputing Applications*", University of Illionis, Chicago.
7. [FRA-10] Francis, Louise, et al, 2010, "*Text Mining Handbook*". *http://www.casact.org/pubs/forum/10spforum/Francis_Flynn.pdf* diakses tanggal 30 Desember 2012.
8. [HAN-06] Han, Jiawei dan Kamber, Micheline, 2006, "*Data Mining: Concepts and Techniques*", Morgan Kaufmann Publishers is an imprint of Elsevier, San Francisco.
9. [HAR-06] Harlian, Milkha, 2006, "*Text Mining*", University of Texas, Austin.
10. [HOO-05] Hooper, Rob dan Paice, Chris, 2005, "*The Lancaster Stemming Algorithm*".
11. [LIU-03] Liu, Dan, et al, 2003, "*Automatic Mood Detection from Acoustic Music Data*", *http://ismir2003.ismir.net/presentations/Liu.pdf* diakses tanggal 1 Desember 2012.

12. [LUZ-06] Luz, S., 2006, "*Machine Learning of Text Categorization*", Information Management - November 29, 2006, Trinity College, Department of Computer Science.
13. [MOO-06] Mooney, R., 2006, "*CS 391L: Machine Learning Text Categorization*", University of Texas, Austin.
14. [MUS-09] Musthafa, Aziz, 2009, "Klasifikasi Otomatis Dokumen Berita Kejadian Berbahasa Indonesia", Universitas Islam Negeri (UIN) Maulana Malik Ibrahim, Malang.
15. [NUG-12]. Nugraha, S., Welly, 2012, "Klasifikasi *E-mail Spam* Berbahasa Inggris Menggunakan Metode *Iterative Dichotomizer Tree*", Program Studi Ilmu Komputer, Universitas Brawijaya, Malang.
16. [POR-80] Porter, M, 1980, "*An Algorithm of Suffix Stripping, Program*, Vol. 14, No. 3, pp, 130-137.
17. [RAM-07] Ramadhan, Rizal, 2007, "Penerapan Pohon Untuk Klasifikasi Dokumen Teks Berbahasa Inggris", Institut Teknologi Bandung, Bandung.
18. [RAG-06] Raghavan, Ratheesh, 2006, "*Study Of The Relationship Of Training Set Size To Error Rate In Yet Another Decision Tree And Random Forest Algorithms*", A Thesis In Computer Science at Texas Tech University.
19. [SUP-00] Supranto, J., 2000, "Statistika: Teori Dan Aplikasi Edisi Keenam", Erlangga, Jakarta.
20. [TRI-09] Triawati, Candra, 2009, "Metode Pembobotan *Statistical Concept Based* untuk Klastering dan Kategorisasi Dokumen Berbahasa Indonesia", IT Telkom, Bandung.
21. [WAH-02] Wahyudi, J., B., 2002, "Dasar-dasar jurnalistik radio dan televisi", Perpustakaan Utan Kayu, Jakarta.
22. [WIB-08] Wibisono, Y., 2008, "*Clustering* Berita Berbahasa Indonesia", Universitas Pendidikan Indonesia, Bandung.
23. [YAN-99] Yang, Y., dan Liu, X., 1999, "*A Re-examination of Text Categorization Methods*", Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval: 42-49.

Lampiran 1
Data *stopword*

No	Term	No	Term	No	Term
1	a	226	Immediate	451	several
2	able	227	immediately	452	shall
3	about	228	importance	453	she
4	above	229	important	454	shed
5	abst	230	in	455	she'll
6	accordance	231	inc	456	shes
7	according	232	indeed	457	should
8	accordingly	233	index	458	shouldn't
9	across	234	information	459	show
10	act	235	instead	460	showed
11	actually	236	into	461	shown
12	added	237	invention	462	showns
13	adj	238	inward	463	shows
14	adopted	239	is	464	significant
15	affected	240	isn't	465	significantly
16	affecting	241	it	466	similar
17	affects	242	itd	467	similarly
18	after	243	it'll	468	since
19	afterwards	244	its	469	six
20	again	245	itself	470	slightly
21	against	246	i've	471	so
22	ah	247	j	472	some
23	all	248	just	473	somebody
24	almost	249	k	474	somehow
25	alone	250	keep	475	someone
26	along	251	keeps	476	somethan
27	already	252	kept	477	something
28	also	253	keys	478	sometime
29	although	254	kg	479	sometimes
30	always	255	km	480	somewhat
31	am	256	know	481	somewhere
32	among	257	known	482	soon
33	amongst	258	knows	483	sorry
34	an	259	l	484	specifically
35	and	260	largely	485	specified
36	announce	261	last	486	specify
37	another	262	lately	487	specifying
38	any	263	later	488	state
39	anybody	264	latter	489	states
40	anyhow	265	latterly	490	still
41	anymore	266	least	491	stop
42	anyone	267	less	492	strongly
43	anything	268	lest	493	sub
44	anyway	269	let	494	substantially
45	anyways	270	lets	495	successfully
46	anywhere	271	like	496	such
47	apparently	272	liked	497	sufficiently
48	approximately	273	likely	498	suggest
49	are	274	line	499	sup
50	aren	275	little	500	sure
51	arent	276	'll	501	t
52	arise	277	look	502	take
53	around	278	looking	503	taken
54	as	279	looks	504	taking
55	aside	280	ltd	505	tell
56	ask	281	m	506	tends
57	asking	282	made	507	th
58	at	283	mainly	508	than
59	auth	284	make	509	thank
60	available	285	makes	510	thanks
61	away	286	many	511	thanx
62	awfully	287	may	512	that
63	b	288	maybe	513	that'll
64	back	289	me	514	thats
65	be	290	mean	515	that've

66	became	291	means	516	the
67	because	292	meantime	517	their
68	become	293	meanwhile	518	theirs
69	becomes	294	merely	519	them
70	becoming	295	mg	520	themselves
71	been	296	might	521	then
72	before	297	million	522	thence
73	beforehand	298	miss	523	there
74	begin	299	ml	524	thereafter
75	beginning	300	more	525	thereby
76	beginnings	301	moreover	526	thered
77	begins	302	most	527	therefore
78	behind	303	mostly	528	therein
79	being	304	mr	529	there'll
80	believe	305	mrs	530	thereof
81	below	306	much	531	therere
82	beside	307	mug	532	theres
83	besides	308	must	533	thereto
84	between	309	my	534	thereupon
85	beyond	310	myself	535	there've
86	biol	311	n	536	these
87	both	312	na	537	they
88	brief	313	name	538	theyd
89	briefly	314	namely	539	they'll
90	but	315	nay	540	theyre
91	by	316	nd	541	they've
92	c	317	near	542	think
93	ca	318	nearly	543	this
94	came	319	necessarily	544	those
95	can	320	necessary	545	thou
96	cannot	321	need	546	though
97	can't	322	needs	547	thoughh
98	cause	323	neither	548	thousand
99	causes	324	never	549	throug
100	certain	325	nevertheless	550	through
101	certainly	326	new	551	throughout
102	co	327	next	552	thru
103	com	328	nine	553	thus
104	come	329	ninety	554	til
105	comes	330	no	555	tip
106	contain	331	nobody	556	to
107	containing	332	non	557	together
108	contains	333	none	558	too
109	could	334	nonetheless	559	took
110	couldnt	335	noone	560	toward
111	d	336	nor	561	towards
112	date	337	normally	562	tried
113	did	338	nos	563	tries
114	didn't	339	not	564	truly
115	different	340	noted	565	try
116	do	341	nothing	566	trying
117	does	342	now	567	ts
118	doesn't	343	nowhere	568	twice
119	doing	344	o	569	two
120	done	345	obtain	570	u
121	don't	346	obtained	571	un
122	down	347	obviously	572	under
123	downwards	348	of	573	unfortunately
124	due	349	off	574	unless
125	during	350	often	575	unlike
126	e	351	oh	576	unlikely
127	each	352	ok	577	until
128	ed	353	okay	578	unto
129	edu	354	old	579	up
130	effect	355	omitted	580	upon
131	eg	356	on	581	ups
132	eight	357	once	582	us
133	eighty	358	one	583	use
134	either	359	ones	584	used

135	else	360	only	585	useful
136	elsewhere	361	onto	586	usefully
137	end	362	or	587	usefulness
138	ending	363	ord	588	uses
139	enough	364	other	589	using
140	especially	365	others	590	usually
141	et	366	otherwise	591	v
142	et-al	367	ought	592	value
143	etc	368	our	593	various
144	even	369	ours	594	've
145	ever	370	ourselves	595	very
146	every	371	out	596	via
147	everybody	372	outside	597	viz
148	everyone	373	over	598	vol
149	everything	374	overall	599	vols
150	everywhere	375	owing	600	vs
151	ex	376	own	601	w
152	except	377	p	602	want
153	f	378	page	603	wants
154	far	379	pages	604	was
155	few	380	part	605	wasn't
156	ff	381	particular	606	way
157	fifth	382	particularly	607	we
158	first	383	past	608	wed
159	five	384	per	609	welcome
160	fix	385	perhaps	610	we'll
161	followed	386	placed	611	went
162	following	387	please	612	were
163	follows	388	plus	613	weren't
164	for	389	poorly	614	we've
165	former	390	possible	615	what
166	formerly	391	possibly	616	whatever
167	forth	392	potentially	617	what'll
168	found	393	pp	618	whats
169	four	394	predominantly	619	when
170	from	395	present	620	whence
171	further	396	previously	621	whenever
172	furthermore	397	primarily	622	where
173	g	398	probably	623	whereafter
174	gave	399	promptly	624	whereas
175	get	400	proud	625	whereby
176	gets	401	provides	626	wherein
177	getting	402	put	627	wheres
178	give	403	q	628	whereupon
179	given	404	que	629	wherever
180	gives	405	quickly	630	whether
181	giving	406	quite	631	which
182	go	407	qv	632	while
183	goes	408	r	633	whim
184	gone	409	ran	634	whither
185	got	410	rather	635	who
186	gotten	411	rd	636	whod
187	h	412	re	637	whoever
188	had	413	readily	638	whole
189	happens	414	really	639	who'll
190	hardly	415	recent	640	whom
191	has	416	recently	641	whomever
192	hasn't	417	ref	642	whos
193	have	418	refs	643	whose
194	haven't	419	regarding	644	why
195	having	420	regardless	645	widely
196	he	421	regards	646	willing
197	hed	422	related	647	wish
198	hence	423	relatively	648	with
199	her	424	research	649	within
200	here	425	respectively	650	without
201	hereafter	426	resulted	651	won't
202	hereby	427	resulting	652	words
203	herein	428	results	653	world

204	heres	429	right	654	would
205	hereupon	430	run	655	wouldn't
206	hers	431	s	656	www
207	herself	432	said	657	x
208	hes	433	same	658	y
209	hi	434	saw	659	yes
210	hid	435	say	660	yet
211	him	436	saying	661	you
212	himself	437	says	662	youd
213	his	438	sec	663	you'll
214	hither	439	section	664	your
215	home	440	see	665	youre
216	how	441	seeing	666	yours
217	howbeit	442	seem	667	yourself
218	however	443	seemed	668	yourselves
219	hundred	444	seeming	669	you've
220	i	445	seems	670	z
221	id	446	seen	671	zero
222	ie	447	self		
223	if	448	selves		
224	i'll	449	sent		
225	im	450	seven		

Lampiran 2

Data *training* dan data *testing* untuk contoh perhitungan manual

Dokumen 1

Dokumen *training*

Kategori : *Angry Song's*

Isi lirik lagu :

Word Screw – Lily Allen

Perhaps one of the most interesting words in the English language today is the word screw.
Out of all the English words that begin with the letter F, screw is the only word referred to as The F word.
Its the one magical word that just by its sound can describe pain, pleasure, hate and love.
screw, as most words in the English language, is derived from German.
The word Flicken which means To strike.
In English, screw falls into many grammatical categories.
As a transitive verb for instance, John screwed Shirley, as an intransitive verb, Shirley screws.
Its meaning is not always sexual.
It can be used as an adjective such as, Johns doing all the screwing work.
As part of an adverb, Shirley talks too screwing much.
As an adverb enhancing an adjective, Shirley is screwing beautiful.
As the object of an adverb, Shirley is screwing beautifully.
As a noun, I dont give a screw.
As part of a word, Abso-screwing-lutly or In-screwing-credible.
And as almost every work in a sentence, screw the screwing screwers.
As you may realise, there are very few words with the versatility of screw.
As in these examples describing situations such as, Fraud: I got screwed at the used car lot.
Aww, screw it, trouble: I guess Im really screwed now.
Dont screw with me buddy.
I dont understand this screwing question.
Who the screw was that.
I dont like what the screw is going on here.
Hes a screw off.
Why dont you go outside and play hide and go screw yourself.?
Im sure you can think of many more examples.
With all of these multipurpose applications, how can anyone be offended when you use this word?
We say use this unique, flexible word more often in your daily speech.
It will identify the qua

Hasil perhitungan frekuensi kata pada dokumen 1

Kata	Frekuensi	Kata	Frekuensi
interest	1	screwer	1
english	4	realis	1
languag	2	versatil	1
todai	1	exampl	1
word	8	situat	5
screw	25	fraud	1
letter	1	car	1
refer	1	lot	1
magic	1	aww	1
sound	1	troubl	1
describ	2	guess	1
pain	1	buddi	1
pleasur	1	understand	2
hate	1	question	1
love	1	go	1
deriv	1	plai	1
german	1	hide	1
flicken	1	multipurpos	1
strike	1	applic	1
fall	1	offend	1
grammat	1	uniqu	1
categori	1	flexibl	1
transit	1	daili	1
verb	2	speech	1
instanc	1	will	1
john	2	identifi	1
shirlei	5	qua	1
intransit	1	beauti	1
mean	1	object	1
sexual	1	beautifulli	1

adject	2	noun	1
work	2	dont	1
adverb	3	abso	1
talk	1	lutli	1
enhanc	1	credibl	1
sentenc	1		

Dokumen 2
Dokumen *training*
Kategori : *Angry Song's*
Isi lirik lagu :

Who the Screw – Jack Of Jill
Who the screw do you think you are
Get out of my hair
who the screw do you think you are
Comin' round here
who the screw who the screw
who the screw do you think you are
Get your comb out of there
combin' out my hair
I'm not like other girls
You can't straighten my curls
I'm not like other girls
You can't straighten my curls
No!
Who the screw you tryin' to be
Get your dog away from me!
What the screw you doing in there
Get your dirty fingers out of my hair
Who who who who
screw screw screw you
I'm free, you'll see
I'm free, you'll see

Hasil perhitungan frekuensi kata pada dokumen 2

Kata	Frekuensi	Kata	Frekuensi
screw	10	curl	2
hair	3	tryin	1
comin	1	dog	1
round	1	dirti	1
comb	1	finger	1
combin	1	free	2
girl	2	ll	2
straighten	2		

Dokumen 3
Dokumen *training*
Kategori : *Fun Song's*
Isi lirik lagu:

Badge Out – Unter Null
Johny's in the basement
Mixing up the medicine
I'm on the pavement
Thinking about the government
The man in the trench coat
Badge out laid off
Says he's got a bad cough
Wants to get it paid off
Look out kid
It's somethin' you did
God knows when
But you're doin' it again
You better duck down the alley way
Lookin' for a new friend
The man in the coonskip cap
In the big pen
Wants eleven dollar bills

You only got ten
Maggie comes fleet foot
Face full of black soot
Talkin' that the heat put
Plants in the bed but
The phone's tapped anyway
Maggie says that many say
They must bust in early May
Orders from the DA
Look out kid
Don't matter what you did
Walk on your tip toes
Don't try, No Doz
Better stay away from those
That carry around a fire hose
Keep a clean nose
Watch the plain clothes
You don't need a weather man
To know which way the wind blows.
Get sick, get well
Hang around an ink well
Ring bell, hard to tell
If anything is goin' to sell
Try hard, get barred
Get back, write Braille
Get jailed, jump bail Join the army, if you failed
Look out kid
You're gonna get hit
But losers, cheaters
Six-time users
Hang around the theaters
Girl by the whirlpool
Lookin' for a new fool Don't follow leaders
Watch the parkin' meters
Ah get born, keep warm
Short pants, romance, learn to dance
Get dressed, get blessed
Try to be a success
Please her, please him, buy gifts
Don't steal, don't lift
Twenty years of schoolin'
And they put you on the day shift
Look out kid
They keep it all hid
Better jump down a manhole
Light yourself a candle
Don't wear sandals
Try to avoid the scandals
Don't wanna be a bum
You better chew gum
The pump don't work
'Cause the vandals took the handles

Hasil perhitungan frekuensi kata pada dokumen 3

Kata	Frekuensi	Kata	Frekuensi
johni	1	sick	1
basement	1	well	2
mix	1	hang	2
medicin	1	ink	1
pavement	1	ring	1
think	1	bell	1
govern	1	hard	2
man	3	goin	1
trench	1	sell	1
coat	1	bar	1
badg	1	write	1
laid	1	braill	1
bad	1	jail	1
cough	1	jump	2
paid	1	bail	1
kid	4	join	1
somethin	1	armi	1

god	1	fail	1
doin	1	gonna	1
better	4	hit	1
duck	1	loser	1
allei	1	cheater	1
lookin	2	time	1
friend	1	user	1
coonskip	1	theater	1
cap	1	girl	1
big	1	whirlpool	1
pen	1	fool	1
eleven	1	follow	1
dollar	1	leader	1
bill	1	parkin	1
ten	1	meter	1
maggi	2	born	1
fleet	1	warm	1
foot	1	short	1
face	1	pant	1
full	1	romanc	1
black	1	learn	1
soot	1	danc	1
talkin	1	dress	1
heat	1	bless	1
plant	1	success	1
bed	1	bui	1
phone	1	gift	1
tap	1	steal	1
bust	1	lift	1
earli	1	twenti	1
order	1	year	1
da	1	schoolin	1
don	9	dai	1
matter	1	shift	1
walk	1	manhol	1
toe	1	light	1
doz	1	candl	1
stai	1	wear	1
carri	1	sandal	1
fire	1	avoid	1
hose	1	scandal	1
clean	1	wanna	1
nose	1	bum	1
watch	2	chew	1
plain	1	gum	1
cloth	1	pump	1
weather	1	work	1
wind	1	vandal	1
blow	1	handl	1

Dokumen 4

Dokumen *training*

Kategori : *Fun Song's*

Isi lirik lagu :

90 Miles – Aterage Image

Well they gave him his orders at Monroe Virginia sayin' Steve you're way behind time This is not 38 this is old 97 you must put her into Spencer on time

Then he turned around and said to his black greasy fireman shovel on a little more coal And when we cross that White Oak Mountain watch old 97 roll

But it's a mighty rough road from Lynchburg to Danville With a line on a three mile grade It was on that grade that he lost his air brakes see what a jump he made

He was goin' down the grade makin' 90 miles an hour his whistle broke into a scream He was found in the wreck with his hand on the throttle A scalded to death by the steam Then the telegram come to Washington station and this is how it read Oh that brave engineer that run old 97 he's a layin' in old Danville dead

So now all you ladies you better take a warnin' from this time on and learn Never speak harsh words to your true lovin' husband He may leave you and never return

Hasil perhitungan frekuensi kata pada dokumen 4

Kata	Frekuensi	Kata	Frekuensi
well	1	goin	1
order	1	makin	1
monro	1	hour	1
virginia	1	whistl	1
sayin	1	broke	1
steve	1	scream	1
time	3	wreck	1
spencer	1	hand	1
turn	1	throttl	1
black	1	scald	1
greasi	1	death	1
fireman	1	steam	1
shovel	1	telegram	1
coal	1	washington	1
cross	1	station	1
white	1	read	1
oak	1	brave	1
mountain	1	engin	1
watch	1	layin	1
roll	1	dead	1
mighti	1	ladi	1
rough	1	better	1
road	1	warnin	1
lynchburg	1	learn	1
danvil	2	speak	1
three	1	harsh	1
mile	2	true	1
grade	3	lovin	1
lost	1	husband	1
air	1	leav	1
brake	1	return	1
jump	1		

Dokumen 5

Dokumen *training*

Kategori : *Love Song's*

Isi lirik lagu :

Heaven – Travis

I'm in the mood for love
Simply because you're near me
Funny, but when you're near me
I'm in the mood for love
Heaven is in your eyes
Bright as the stars we're under
Oh, is it any wonder
That I'm in the mood for love?
Why stop to think of whether
This little dream might fade?
We've put our hearts together
Now we are one, I'm not afraid
And if there's a cloud above
If it should rain, we'll let it
But for tonight forget it
I'm in the mood for love
Oh yeah
Why stop to think of whether
This little dream might fade?
We've put our hearts together
Now we are one, I'm not afraid
And if there's a cloud above
If it should rain, we'll let it
But, for tonight, forget it
Cause I'm in the mood for love
I'm in the mood for love
For love, for love...

Hasil perhitungan frekuensi kata pada dokumen 5

Kata	Frekuensi	Kata	Frekuensi
mood	6	fade	2
love	8	ve	2
simpli	1	heart	2
funni	1	afraid	2
heaven	1	cloud	2
ey	1	rain	2
bright	1	ll	2
star	1	tonight	2
wonder	1	forget	2
dream	2	yeah	1

Dokumen 6

Dokumen *training*

Kategori : *Love Song's*

Isi lirik lagu :

Baby Love You – Otis Redding

I want you the right way
I want you
But I want you to want me too
Want you to want me, baby love you love you love you
Just like I want you
I give you all the love I want in return sweet daring. Pure happiness is all I cave.
It's too bad, It's just too sad
You don't want me now
But I'm gonna change your mind
Someway,somehow,oh baby love you this one way love is just a fantasy, oh sugar
To share is precious, pure and fair
Don't play with something you should cherish for life,oh baby love you
Don't you wanna care
Ain't it lonely out there

Hasil perhitungan frekuensi kata pada dokumen 6

Kata	Frekuensi	Kata	Frekuensi
babi	3	somewai	1
love	7	repeat	1
return	1	fantasi	1
sweet	1	sugar	1
dare	1	share	1
pure	2	preciou	1
happi	1	fair	1
cave	1	plai	1
bad	1	cherish	1
sad	1	life	1
don	3	wanna	1
gonna	1	care	1
chang	1	ain	1
mind	1	lone	1

Dokumen 7

Dokumen *training*

Kategori : *Sad Song's*

Isi lirik lagu :

Without You - George Jones

How do I,
Get through one night without you?
If I had to live without you,
What kind of life would that be?
Oh, I...
I need you in my arms, need you to hold,
You're my world, my heart, my soul,
If you ever leave,
Baby you would take away everything good in my life,
And tell me now
How do I live without you?

I want to know,
How do I breathe without you?
If you ever go,
How do I ever, ever survive?
How do I, how do I, oh how do I live?
Without you,
There'd be no sun in my sky,
There would be no love in my life,
There'd be no world left for me.
And I,
Baby I don't know what I would do,
I'd be lost if I lost you,
If you ever leave,
Baby you would take away everything real in my life,
And tell me now, How do I live without you? I want to know,
How do I breathe without you?
If you ever go, How do I ever, ever survive?
How do I, how do I, oh how do I live? Please tell me baby,
How do I go on? If you ever leave,
Baby you would take away everything,
I need you with me,
Baby don't you know that you're everything,
Good in my life?
And tell me now, How do I live without you,
I want to know,
How do I breathe without you?
If you ever go,
How do I ever, ever survive?
How do I, how do I, oh how do I live?
How do I live without you?
How do I live without you baby?

Hasil perhitungan frekuensi kata pada dokumen 7

Kata	Frekuensi	Kata	Frekuensi
night	1	good	2
live	9	breath	3
kind	1	surviv	3
life	5	sun	1
arm	1	sky	1
hold	1	love	1
heart	1	left	1
soul	1	don	2
leav	3	lost	2
babi	7	real	1

Dokumen 8

Dokumen *training*

Kategori : *Sad Song's*

Isi lirik lagu :

Hurt - Iron Butterfly

I hurt myself today To see if I still feel I focus on the pain The only thing that's real The needle tears a hole The old familiar sting Try to kill it all away But I remember everything What have I become My sweetest friend Everyone I know goes away In the end And you could have it all My empire of dirt I will let you down I will make you hurt I wear this crown of thorns Upon my liar's chair

Full of broken thoughts I cannot repair Beneath the stains of time The feelings disappear You are someone else I am still right here What have I become My sweetest friend Everyone I know goes away In the end And you could have it all My empire of dirt I will let you down I will make you hurt If I could start again A million miles away I would keep myself I would find away

Hasil perhitungan frekuensi kata pada dokumen 8

Kata	Frekuensi	Kata	Frekuensi
hurt	4	thorn	1
todai	1	liar	1
feel	2	chair	1
focu	1	http	1
pain	1	metrolyr	1
thing	1	lyric	1
real	1	johnni	1
needl	1	cash	1
tear	1	html	1

hole	1	full	1
familiar	1	broken	1
sting	1	thought	1
kill	1	repair	1
rememb	1	beneath	1
sweetest	2	stain	1
friend	2	time	1
empir	2	disappear	1
dirt	2	start	1
will	4	mile	1
wear	1	find	1
crown	1		

Dokumen 9

Dokumen *testing*

Kategori : Tidak Berkategori

Isi lirik lagu:

I Hate You – Unter Null

Every time we lie awake
After every hit we take
Every feeling that I get
But I haven't missed you yet
Every roommate kept awake
By every sigh and scream we make
All the feelings that I get
But I still don't miss you yet
Only when I stop to think about it
I hate everything about you
Why do I love you?
I hate everything about you
Why do I love you?
Every time we lie awake
After every hit we take
Every feeling that I get
But I haven't missed you yet
Only when I stop to think about it
I hate everything about you
Why do I love you?
I hate everything about you
Why do I love you?
Only when I stop to think about you,
I know
Only when you stop to think about me,
do you know?
I hate everything about you
Why do I love you?
You hate everything about me
Why do you love me?
I hate
You hate
I hate
You love me
I hate everything about you
Why do I love you?

Hasil perhitungan frekuensi kata pada dokumen 9

Kata	Frekuensi	Kata	Frekuensi
time	2	roommat	1
lie	2	sigh	1
awak	3	scream	1
hit	2	don	1
feel	3	hate	10
haven	2	love	8
miss	2		

Dokumen 10
Dokumen *testing*
Kategori : Tidak Berkategori
Isi lirik lagu:

Number One For Me– Maher Zain
I was a foolish little child
Crazy things I used to do
And all the pain I put you through
Mama now I'm here for you
For all the times I made you cry
The days I told you lies
Now it's time for you to rise
For all the things you sacrificed
Oh, if I could turn back time rewind
If I could make it undone
I swear that I would
I would make it up to you
Mum I'm all grown up now
It's a brand new day
I'd like to put a smile on your face every day
Mum I'm all grown up now
And it's not too late
I'd like to put a smile on your face every day
And now I finally understand
Your famous line
About the day I'd face in time
'Cause now I've got a child of mine
And even though I was so bad
I've learned so much from you
Now I'm trying to do it too
Love my kid the way you do
You know you are the number one for me
Oh, oh, number one for me
There's no one in this world that can take your place
Oh, I'm sorry for ever taking you for granted, ooh
I will use every chance I get
To make you smile, whenever I'm around you
Now I will try to love you like you love me
Only God knows how much you mean to me
You know you are the number one for me
Oh, oh, number one for me

Hasil perhitungan frekuensi kata pada dokumen 10

Kata	Frekuensi	Kata	Frekuensi
foolish	1	smile	3
child	2	face	3
crazi	1	late	1
thing	2	final	1
pain	1	understand	1
mama	1	famou	1
time	4	ve	2
cry	1	mine	1
dai	5	bad	1
told	1	learn	1
li	1	love	3
rise	1	kid	1
sacrif	1	number	4
turn	1	place	1
rewind	1	grant	1
undon	1	ooh	1
swear	1	will	2
mum	2	chanc	1
grown	2	god	1
brand	1		

Dokumen 11
Dokumen *testing*
Kategori : Tidak Berkategori
Isi lirik lagu:

Call Me – Carl Rae Japsen
I threw a wish in the well,
Don't ask me, I'll never tell
I looked to you as it fell,
And now you're in my way
I'd trade my soul for a wish,
Pennies and dimes for a kiss
I wasn't looking for this,
But now you're in my way
Your stare was holdin'
Ripped jeans, skin was showin'
Hot night, wind was blowin'
Where do you think you're going, baby?
Hey, I just met you,
And this is crazy,
But here's my number,
So call me, maybe!
It's hard to look right
At you baby,
But here's my number,
You took your time with the call,
I took no time with the fall
You gave me nothing at all,
But still, you're in my way
I beg, and borrow and steal
Have foresight and it's real
I didn't know I would feel it,
But it's in my way
Your stare was holdin',
Ripped jeans, skin was showin'
Hot night, wind was blowin'
Where you think you're going, baby?
Hey, I just met you,
And this is crazy,
But here's my number,
So call me, maybe!
It's hard to look right
At you baby,
But here's my number,
So call me, maybe!
Hey, I just met you,
And this is crazy,
But here's my number,
So call me, maybe!
And all the other boys,
Try to chase me,
But here's my number,
So call me, maybe!
Before you came into my life
I missed you so bad
I missed you so bad
I missed you so, so bad
Before you came into my life
I missed you so bad
And you should know that
I missed you so, so bad (bad, bad)
It's hard to look right
At you baby,
But here's my number,
So call me, maybe!
Hey, I just met you,
And this is crazy,
But here's my number,
So call me, maybe!
And all the other boys,
Try to chase me,
But here's my number,

So call me, maybe!
Before you came into my life
I missed you so bad
I missed you so bad
I missed you so, so bad
Before you came into my life
I missed you so bad
And you should know that
So call me, maybe!

Hasil perhitungan frekuensi kata pada dokumen 11

Kata	Frekuensi	Kata	Frekuensi
threw	1	go	2
well	1	babi	5
don	1	hei	5
ll	1	met	5
look	1	crazi	5
fell	1	number	11
trade	1	call	13
soul	1	hard	3
penni	1	boi	3
dime	1	chase	3
kiss	1	time	2
wasn	1	fall	1
stare	2	beg	1
holdin	2	borrow	1
rip	2	steal	1
jean	2	foresight	1
skin	2	real	1
showin	2	didn	1
hot	2	feel	1
night	2	life	4
wind	2	miss	9
blowin	2	bad	11

Dokumen 12
Dokumen *testing*
Kategori : Tidak Berkategori
Isi lirik lagu:

Jar Of Hearts – Christina Perry
No, I can't take one more step towards you
'Cause all that's waiting is regret
Don't you know I'm not your ghost anymore
You lost the love I loved the most
I learned to live half alive
And now you want me one more time
And who do you think you are?
Runnin' 'round leaving scars
Collecting your jar of hearts
And tearing love apart
You're gonna catch a cold
From the ice inside your soul
So don't come back for me
Who do you think you are?
I hear you're asking all around
If I am anywhere to be found
But I have grown too strong
To ever fall back in your arms
And I've learned to live half alive
And now you want me one more time
And who do you think you are?
Runnin' 'round leaving scars
Collecting your jar of hearts
And tearing love apart
You're gonna catch a cold
From the ice inside your soul
So don't come back for me
Who do you think you are?
And it took so long just to feel alright
Remember how to put back the light in my eyes

I wish I had missed the first time that we kissed
'Cause you broke all your promises
And now you're back
You don't get to get me back
And who do you think you are?
Runnin' 'round leaving scars
Collecting your jar of hearts
And tearing love apart
You're gonna catch a cold
From the ice inside your soul
So don't come back for me
Don't come back at all
And who do you think you are?
Runnin' 'round leaving scars
Collecting your jar of hearts
Tearing love apart
You're gonna catch a cold
From the ice inside your soul
Don't come back for me
Don't come back at all
Who do you think you are?
Who do you think you are?
Who do you think you are?

Hasil perhitungan frekuensi kata pada dokumen 12

Kata	Frekuensi	Kata	Frekuensi
step	1	catch	4
wait	1	cold	4
regret	1	ic	4
don	8	insid	4
ghost	1	soul	4
lost	1	hear	1
love	6	grown	1
learn	2	strong	1
live	2	fall	1
half	2	arm	1
aliv	2	ve	1
time	3	long	1
runnin	4	feel	1
round	4	alright	1
leav	4	rememb	1
scar	4	light	1
collect	4	ey	1
jar	4	miss	1
heart	4	kiss	1
tear	4	broke	1
apart	4	promis	1
gonna	4		