22015/H/05

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK PENGKLASIFIKASIKAN DOKUMEN BERBASIS ORACLE TEXT 10 DENGAN ALGORITMA SUPPORT VECTOR MACHINE

TUGAS AKHIR

RSIF
005.1
Dew
p-1
2005

Disusun Oleh :

NUGRAHA DEWATA
NRP 5197.100.076

PERPUSTAKAAN
ITS

Tgl. Terima	5-4-2005
Terima Dari	H
No. Agenda Prp.	221785

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2005

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK PENGKLASIFIKASIKAN DOKUMEN BERBASIS ORACLE TEXT 10 DENGAN ALGORITMA SUPPORT VECTOR MACHINE

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan

Untuk Memperoleh Gelar Sarjana Komputer

Pada

Jurusan Teknik Informatika

Fakultas Teknologi Informasi

Institut Teknologi Sepuluh Nopember

Surabaya

Mengetahui/Menyetujui

Dosen Pembimbing

7/2 '05

Rully Soelaiman, S. Kom., M.Kom.

NIP. 132.085.802

Surabaya, Januari 2005

ABSTRAK

Abstrak – Dalam tugas akhir ini akan dibahas mengenai pembuatan aplikasi text classification dengan algoritma Support Vector Machines yang dibangun diatas platform Oracle Text 10.1. Algoritma SVM sangat dikenal karena kemampuannya dalam pengenalan pola dan klasifikasi terbukti memiliki tingkat akurasi yang sangat tinggi [Bur-1998]. Ide dasar dari algortima support vector machine adalah untuk menemukan hyperplane optimal yang akan mengklasifikasikan data dengan benar dengan cara memisahkan nilai – nilai diantara dua kelas sejauh mungkin. Pencarian hyperplane yang optimal ini harus dapat juga meminimalkan kesalahan klasifikasi terhadap data training dan data testing yang belum pernah di uji coba sebelumnya.

Metodologi yang digunakan dalam pembuatan tugas akhir ini terdiri atas beberapa tahapan. Tahap pertama adalah proses loading dokumen kedalam database. Tahap berikutnya adalah preprocessing dokumen agar dapat diproses dalam algoritma klasifikasi. Dilanjutkan dengan proses indexing, dan dilakukan proses trainning dengan menggunakan SVM classifier, hingga didapatkannya rule – rule pengklasifikasi. Rule – rule ini nantinya juga akan diindeks. Tahap terakhir adalah uji coba terhadap rule, dengan kumpulan dokumen test.Keseluruhan proses ini dijalankan diatas platfrom Oracle Text 10.1.

Uji coba dan evaluasi dilakukan dengan menggunakan data UseNet Collection (20NG), yang berisi 20.000 dokumen newsgroups yang telah dikelompokkan dalam 20 kelompok yang berbeda. Kemampuan SVM classifier ini juga akan dibandingkan dengan aplikasi text classification yang pernah dibuat sebelumnya [Pur-2004]. Dari hasil uji coba di dapatkan bahwa SVM Classifier memiliki kemampuan yang lebih baik dari pada decission tree classifier dengan perbedaan kemampuan klasifikasi rata – rata sebesar ± 20 %. Sedangkan kemampuan rata – rata klasifikasi dari SVM Classifier sendiri mencapai ± 90 %.

KATA PENGANTAR

Alhamdulillah, segala puji dan syukur semata ditujukan ke hadirat Allah SWT yang telah memberikan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir berjudul "**Perancangan dan pembuatan perangkat lunak pengklasifikasi dokumen berbasis Oracle Text 10 dengan algoritma Support Vector Machine** ".

Tugas Akhir ini merupakan salah satu syarat untuk menyelesaikan program strata satu (S-1) pada jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Tiada gading yang tak retak, Penulis menyadari bahwa masih banyak kekurangan dalam pengerjaan tugas akhir ini, oleh sebab itu, penulis selalu membuka diri untuk menerima saran, kritik dan masukan dari berbagai pihak. Dan penulis berharap semoga apa yang telah penulis kerjakan dapat bermanfaat dan dapat dikembangkan lebih lanjut untuk perkembangan ilmu pengetahuan dan teknologi dimasa mendatang.

Surabaya, Februari 2005

Penulis

UCAPAN TERIMA KASIH

Penulis menyadari bahwa Tugas Akhir ini tidak lepas dari bantuan banyak pihak, maka dalam kesempatan ini, penulis ingin menyampaikan penghormatan dan ucapan terima kasih kepada pihak-pihak yang telah memberi bantuan baik itu berupa moril maupun material secara langsung maupun tidak langsung, kepada :

1. Bapak Prof. Dr. Ir. Arif Djunaidy, M.Sc, selaku Ketua Dekan Fakultas Teknologi Informasi, ITS.
2. Bapak Yudhi Purwananto, S.Kom, M.Kom, selaku Ketua Jurusan Teknik Informatika, FTIF, ITS.
3. Bapak Rully Soelaiman, S.Kom, M.Kom, selaku Dosen Pembimbing, yang tidak pernah bosan mengingatkan penulis agar tetap semangat mengerjakan tugas akhir, memberikan bimbingan, petunjuk, dan nasehat sejak penulis masuk kuliah sampai tugas akhir selesai.
4. Bapak dan Ibu dosen Jurusan Teknik Informatika, yang telah memberikan ilmunya selama penulis menempuh masa perkuliahan.
5. Staf dan karyawan Jurusan Teknik Informatika – FTIf – ITS yang telah membantu penulis untuk urusan administrasi akademik selama masa perkuliahan.
6. Bapak dan Ibu tercinta, yang telah dengan sabar dan ikhlas mendidik, dan membesarkan penulis.
7. Mas Didit & Mbak Nunik, mas Agung dan Lila, saudara – saudaraku yang menjadi sumber inspirasi, teman curhat, diskusi, dan belajar memaknai hidup,

juga Fikri ponakanku tersayang yang selalu membawa keceriaan dan gelak tawa bagi penulis di rumah, terutama pada saat capek mengerjakan tugas akhir.

8. *My very best friend* Ilham, teman berbagi kebahagiaan dan kesedihan, semoga segera dipertemukan dengan jodohnya.

9. Nuning'99, terima kasih atas semua bantuannya dalam mengerjakan tugas akhir, dorongan dan semangat, juga kerelaan atas tugas akhirnya yang saya jadikan sebagai bahan perbandingan. Juga terima kasih buat Ochin, atas ijin yang diberikan buat nuning, kapan nih undangannya ?

10. Teman – teman seperjuangan TA judul pertama, Hera, Hedrig, Daning + Yazid (terimakasih mau bantuin ngangkut komputer dari kantor ke kampus), Adhita, dan Tita, terimakasih atas semuanya, walau kalian meninggalkan ku lebih dulu.

11. Temen – teman seperjuangan TA judul kedua, Gershom 'the lab man', Andry 'Gondronk' Hussain, Daru, Qayim, Lukman, Inggar, Yuniawan, Dody Dewa Dody Rachmat, Doddy Madura, dan semua 'juru kunci' 97, Ray'99, Kecu (nama aslinya siapa sih?), Indy, Awe, Yudha'97, Dody Widi, ayo rek lulus bareng !.

12. Penghuni Lab. Komputing, tempat penulis menjalani "masa karantina", Eddy'01 terimakasih sudah dibantu ngeprint buku TA, Robby'01 terima kasih atas PLSql-nya, dan semua Admin Lab. Komputing, juga Lab. Sistem Informasi.

13. Seluruh Teman – teman C0D, terima kasih atas dukungannya selama ini Dio, Eddy Fahmi, Dedik, David, Siang, Vicka, Fajar, Sofiah, Tutik, Danie, Winnie, Pinjie, Dian, Eddy Yuniar, Nopin, Nova, Dody Dewa, Ochin dan semuanya yang tidak dapat disebutkan satu persatu.

14. Bapak M. Nur Salim, Teman – teman dosen P2KB Unesa, dan karyawan, terima kasih sudah memberikan kesempatan buat penulis untuk mengaktualisasikan diri di dunia pendidikan.

15. Teman – teman Beesoft, Pak Fauzy (saya belajar banyak dari anda), bu Yuni, Yuhana + mas Novri, Fahrul'99, Asmunin'97, Pak Yudi'97 + Bu Urina, dan Mbak Willy semoga semua impian dan cita – citanya tercapai, kapan nih mroyek lagi ?

16. Teman – teman IWORKS.ORG + COSMIC : Agus, Diorama, Hussain, Eka, Rachmat'99, Mbak Meme, Ardian, Lavdate, Taufan, Danang, Ufan, Erwin, dan mas Eric, dan *special thanks* buat Irvan !

17. Teman – teman SEVIMA, henky'97, Halim'97, Yahya, Dul, Bambang dan Yudi.

18. Teman – teman online, baik di yahoo messenger, ICQ, MIRC, maupun friendster, semuanya tanpa terkecuali.

19. Serta rekan-rekan lain yang tidak dapat disebutkan satu-persatu yang telah membantu penulis mulai dari awal kuliah hingga tersusunnya tugas akhir ini.

DAFTAR ISI

ABSTRAK iii
KATA PENGANTAR iv
UCAPAN TERIMA KASIH v
DAFTAR ISI viii
DAFTAR GAMBAR xi
DAFTAR TABEL xiii
DAFTAR TABEL xiii
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Tujuan dan Manfaat 5
1.3 Permasalahan 5
1.4 Batasan Masalah 6
1.5 Metodologi Tugas Akhir 7
1.6 Sistematika Penulisan 8
BAB 2 DASAR TEORI 10
2.1 Information retrieval 10
2.2 preprocessing 13
2.2.1 Analisa leksikal 14
2.2.2 Penghilangan Stopword 14
2.2.3 Stemming 15
2.2.4 Pemilihan Kata Terindeks untuk Menentukan Grup Kata 22
2.2.5 Kontruksi Struktur Kategorisasi Kata (*Thesaurus*) 23
2.3 Pengindeksan(*indexing*) 26
2.4 *Machine learning* dalam Pengklasifikasian dokumen 29
2.4.1 SVM CLASSIFIER 30
2.4.2 VC Dimension 32
2.4.3 Support Vector Machine 33
2.4.4 Keunggulan SVM Untuk Pengklasifikasian teks. 35
BAB 3 ORACLE TEXT 10.1 37

3.1 Gambaran umum oracle text 37
3.2 Perbedaan Oracle Text 9.2 dengan Oracle Text 10.1 38
3.2.1 *Decision Trees Classifier* dalam Oracle Text 9.2 39
3.2.2 Classifier dalam Oracle Text 10.1 41
3.3 langkah – langkah menggunakan oracle text 42
3.4 Mempopulasi tabel dokumen 43
3.4.1 SQL INSERT 43
3.4.2 SQL*LOADER 44
3.4.3 Prosedur PL/SQL Insert dengan EMPTY_CLOB atau EMPTY_BLOB 47
3.5 Membuat Indeks pada tabel dokumen 48
3.5.1 Tipe Indeks pada Oracle Text 48
3.5.2 Langkah-Langkah Membuat Indeks 55
3.5.3 Jenis – jenis Index Preference 56
3.6 Melakukan proses *training* dengan CTX_CLS.TRAIN 72
3.7 Supervised Classification berbasis SVM 76
BAB 4 PERANCANGAN DAN IMPLEMENTASI PERANGKAT LUNAK 79
4.1 Perancangan Perangkat Lunak 79
4.1.1 Perancangan data 80
4.1.2 perancangan proses 82
4.1.3 perancangan antar muka 90
4.2 Implementasi Perangkat Lunak 91
4.2.1 Lingkungan Implementasi 91
4.2.2 Implementasi data 92
4.2.3 IMPLEMENTASI PROSES 97
4.2.4 IMPLEMENTASI ANTAR MUKA PENGGUNA 99
BAB 5 UJICOBA DAN EVALUASI 104
5.1 Lingkungan Pengujian 104
5.2 Data Uji Coba 104
5.3 Pelaksanaan Uji Coba 106
5.3.1 Perbandingan Uji Coba 106

5.3.2 Uji Pengaruh Parameter Index 116
5.3.3 Evaluasi Uji Coba 119
BAB 6 KESIMPULAN DAN SARAN 124
6.1 Kesimpulan 124
6.2 Saran 125
DAFTAR PUSTAKA ix
LAMPIRAN A 1
PREFERENCE BASIC LEXER PADA ORACLE TEXT 10.1 1
LAMPIRAN B 1
INSTALASI DATABASE ORACLE 10 G 1
LAMPIRAN C 1
CONTOH DAN TAMPILAN PEMROSESAN DOKUMEN DENGAN ORACLE TEXT 10.1 1

DAFTAR GAMBAR

Gambar 1.1 Contoh klasifikasi support vector machine 3
Gambar 2.1 Gambaran Umum Sistem Informasi Temu Kembali 12
Gambar 2.2 Bagan Tahapan Teks Pre-processing 13
Gambar 2.3 Merepresentasikan teks dokumen sebagai vector 30
Gambar 2.4 (a) Hyperplane pemisah dengan margin kecil. (b) Hyperplane pemisah dengan margin besar 31
Gambar 2.5 Kombinasi VC Dimension untuk R^2 33
Gambar 3.1 Konsep dasar SVM dalam ruang berdimensi dua 42
Gambar 4.1 Bagan proses aplikasi text classification 80
Gambar 4.2 Process Tree 83
Gambar 4.3 Context Diagram (DFD level 0) 84
Gambar 4.4 Overview Diagram (DFD level 1) 85
Gambar 4.5 DFD level 2 untuk proses load document 86
Gambar 4.6 DFD level 2 untuk proses create project 87
Gambar 4.7 DFD level 2 untuk proses Trainning 88
Gambar 4.8 DFD level 2 untuk proses Testing 88
Gambar 4.9 DFD level 2 untuk proses Reporting 89
Gambar 4.10 Hirarki menu aplikasi 90
Gambar 4.11 Model fisik basis data 92
Gambar 4.12 FormUtama dari aplikasi 99
Gambar 4.13Form Koneksi Ke database 100
Gambar 4.14 Form Loading Document 101
Gambar 4.15 Form Create New Project (index setting) 102
Gambar 4.16 Form Create New Project (index setting) 102
Gambar 5.1 Hasil uji coba dan perbandingan klasifikasi data dengan 3-fold validation, apply stemming, penghilangan stopwords dan penggunaan thesaurus 107

Gambar 5.2 Hasil uji coba dan perbandingan klasifikasi data dengan 3-fold validation, penghilangan stopwords 111
Gambar 5.3 Tabel perbandingan rata- rata kemampuan klasifikasi antara Dec. Tree classifier dengan SVM classifier, dengan 3-fold data validation dan penggunaan stemming 113
Gambar 5.4 Hasil uji coba dan perbandingan klasifikasi data dengan 10-fold validation, apply stemming, penghilangan stopwords dan penggunaan thesaurus 114
Gambar 5.5 Hasil uji coba Perbadingan penghilangan stop word dan tidak dihilangkan 117
Gambar 5.6 Hasil uji coba Perbadingan penggunaan stemming dan tanpa stemming 119
Gambar 5.7 Evaluasi perbandingan hasil uji coba pada dec. tree classifier dengan svm classifier untuk 3-fold data validation 120
Gambar 5.8 Evaluasi hasil perbandingan uji coba pada dec. tree classifier dengan svm classifier untuk 10-fold data validation 120
Gambar 5.9 Evaluasi Hasil uji coba pengaruh konfigurasi parameter indeks..... 121
Gambar 5.10 Evaluasi Hasil uji coba pengaruh konfigurasi parameter trainning 123

DAFTAR TABEL

Tabel 3.1 Index Type Overview 48
Tabel 3.2 Daftar Preference dalam Oracle Text 55
Tabel 3.3 Daftar Datastore Preference 57
Tabel 3.4 List Preference Filter 58
Tabel 3.5 List Preference Lexer 59
Tabel 3.6 Atribut BASIC_WORDLIST 61
Tabel 3.7 Atribut BASIC_STORAGE 65
Tabel 3.8 Tipe-tipe grup pada SECTION_GROUP 69
Tabel 3.9 Daftar Preference Pada SVM_Classifier 75
Tabel 5.1 Data UseNet Collection menurut subyek 105
Tabel 5.2 Deskripsi data set UseNet Collection 105
Tabel 5.3 Tabel perbandingan rata- rata kemampuan klasifikasi antara Dec. Tree classifier dengan SVM classifier, dengan 3-fold data validation, penggunaan stemming, penghilangan stopword dan penggunaan thesaurus 108
Tabel 5.4 Tabel perbandingan rata- rata kemampuan klasifikasi antara Dec. Tree classifier dengan SVM classifier, dengan 3-fold data validation, penghilangan stopword 110
Tabel 5.5 Tabel perbandingan rata- rata kemampuan klasifikasi antara Dec. Tree classifier dengan SVM classifier, dengan 3-fold data validation, dan penggunaan stemming 111
Tabel 5.6 Tabel perbandingan rata- rata kemampuan klasifikasi antara Dec. Tree classifier dengan SVM classifier, dengan10-fold data validation, penggunaan stemming, penghilangan stopword dan penggunaan thesaurus 115
Tabel 5.7 Perbandingan uji coba dengan penghilangan stopword 116
Tabel 5.8 Perbandingan uji coba dengan penggunan stemming 118

BAB I

PENDAHULUAN

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG

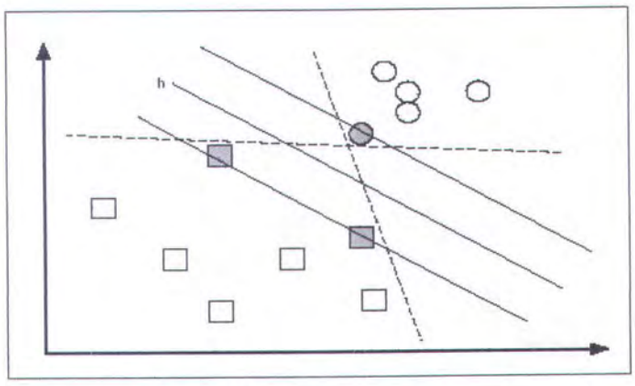
Makin banyaknya dokumen dalam format elektronik dan setiap hari selalu bertambah dan mudah diakses, dokumen – dokumen tersebut misalnya berupa surat elektronik (e-mail), publikasi akademis, dan jurnal atau dokumen dalam sebuah situs berita misalnya.

Dokumen – dokumen semacam ini menyimpan informasi dalam jumlah yang luar biasa banyaknya, namun bagaimanapun juga melakukan pencarian informasi dari kumpulan dokumen semacam ini diperlukan metode yang baik. Pada sistem klien email biasanya digunakan metode penyaringan email. Dan pada situs-situs akademis biasanya disediakan fasilitas untuk pencarian dokumen atau jurnal serta menampilkan daftar dokumen yang terorganisasi dalam bentuk hirarki (misalkan berdasarkan subyek, penulis dan sebagainya)

Namun untuk melakukan itu semua secara manual adalah merupakan hal yang sulit dan merepotkan, yang sebenarnya bisa digantikan dengan sistem pengklasifikasian otomatis, dimana tingkat akurasi dari sistem yang digunakan akan menunjukkan efektifitas penggunaannya.

Support Vector Machine merupakan salah satu teknik yang digunakan dalam pembelajaran mesin (*machine learning)* berdasarkan teori pembelajaran statisikal yang diperkenalkan oleh Vapnik [Vap-1995] dan dapat digunakan untuk pengklasifikasian teks yang di ujicoba pertama kali oleh Joachims [Joa-1998],

untuk melakukan percobaan klasifikasi teks biner mengunakan SVM. SVM menghasilkan nilai error yang lebih kecil daripada teknik klasifikasi teks yang lain. 'Yang' kemudian mengikuti dengan eksperimennya pada *data set* yang sama, namun menggunakan versi Nive Bayes (NB) dan kNN dan masih menemukan bahwa SVM memiliki kemapuan yang sedikit lebih baik. 'Yang' juga menemukan bahwa SVM linear memiliki kemampuan sebaik yang dimiliki oleh SVM polynomial dan RBF. Kedua paper ini menggunakan SVM untuk klasifikasi teks biner. Konsep dasar penggunaan SVM untuk permasalahan klasifikasi adalah, bagaimana mencari batas pemisah (*separating hyperplane*) yang optimal diantara contoh-contoh data set positif dan negatif. Dan batas optimal ini didefinisikan sebagai nilai yang dapat memberikan margin maksimal antara data – data contoh pelatihan yang terdekat dengan batas pemisah. Dan kemudian kelompok vektor yang terdekat dengan batas pemisah ini disebut sebagai vektor pendukung atau *suppport vector.* Sebagai visualisasi dapat dilihat dalam gambar 1.1 dibawah dengan garis **h** sebagai *separating hyperplane*. Ketika batas pemisah sudah ditemukan maka data-data baru (selain data pelatihan) dapat dimasukkan dan diklasifikasikan hanya dengan memeriksa disisi mana (dari garis pemisah) data tersebut akan berada.



Gambar 1.1 Contoh klasifikasi support vector machine

Dalam tugas akhir kali ini akan ditawarkan metode pengklasifikasian dokumen berbasis teks (text classification) dengan menggunakan metode Support Vector Machine (SVM). Metode *Support Vector Machine* biasa berusaha meng-induksi fungsi keputusan dalam proses pembelajarannya, sedangkan *Support Vector Machine* (SVM) hanya menggunakan sebagian dari data pelatihan dan berusaha untuk meminimalkan kesalahan klasfikasi hanya untuk kumpulan data tersebut. Beberapa contoh proses yang menggunakan metode *Text Classification* adalah sebagai berikut :

1. *Relevance* Feedback : langkah ini merupakan teknik yang standar digunakan di dalam *free-text Information reterival*. Dimana pengguna menandai dokumen yang akan dicari dengan menggunakan *query* sebagai kriteria yang diharapkan atau tidak diharapkan. Langkah ini juga mencakup proses pelatihan klasifikasi teks, dimana dokumen yang ada dalam database menjadi bahan ujicobanya.
2. *News Filtering* : Setiap hari sejumlah besar artikel berita dibuat dan di publikasikan. Sebuah situs berita dapat memberikan pilihan artikel yang paling diminati oleh setiap pengguna setiap harinya berdasarkan hasil dari

seperti format file - file aplikasi microsoft office, file - file Adobe PDF TM, HTML dan XML.

Oracle teks juga menyediakan fitur - fitur multi bahasa yang mampu melakukan proses pencarian dalam dokumen berbahasa barat (Inggris, Perancis, Jerman dan sebagainya), Jepang, Korea, dan Cina. Sebagai bagian dari oracle 10g, oracle text terintegrasi secara transparan dan mampu memberikan keuntungan dari fitur - fitur yang diharapkan dari aplikasi enterprise, seperti :

- partisi data (untuk throughput dan ketersediaan data yang tinggi)
- aplikasi clustering yang nyata dan server paralel (kemampuan sever yang tinggi)
- optimasi query
- ketersediaan perangkat dan lingkungan pembangun
- Administrasi dan pengaturan
- serta perangkat keamanan yang sudah terintegrasi

1.2 Tujuan dan Manfaat

Tujuan dari pembuatan Tugas Akhir ini adalah untuk mengembangkan aplikasi pengklasifikasian dokumen multiformat berbahasa inggris secara otomatis dengan menggunakan algoritma *Support Vector Machine* (SVM), berbasis Oracle 10g dan Oracle Text 10.1 sebagai *information retreival engine*.

1.3 Permasalahan

Adapun permasalahan dalam pembuatan tugas akhir ini adalah sebagai berikut ini.

1. Bagaimana melakukan proses preprocessing dokumen diatas skema oracle 10g..
2. Bagaimana menerapkan algoritma *Support Vector Machine* (SVM) dalam sistem berbasis Oracle 10g.
3. Bagaimana membangun sebuah aplikasi yang berplatform java dengan menggunakan fasilitas yang ada pada Oracle Text 10.1
4. Bagaimana melakukan uji coba sistem untuk mendapatkan hasil analisa kemapuan aplikasi dengan benar.

1.4 BATASAN MASALAH

Batasan masalah yang diterapkan dalam pembuatan tugas akhir ini adalah sebagai berikut :

1. Sistem ini akan menggunakan hampir semua fitur dari Oracle 10g untuk diimplementasikan dalam *engine* pengklasifikasian. Termasuk Oracle Text dengan content manajemennya yang juga terdapat dalam Oracle 10g.
2. Dokumen yang digunakan adalah dokumen-dokumen berbahasa Inggris dengan format yang biasa digunakan pada paper, misalnya berekstensi .doc, .pdf, .rtf dan lain-lain.
3. Oracle 10g dibangun di atas platform Windows 2000 Professional atau Windows Advanced Server.

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

1.5 METODOLOGI TUGAS AKHIR

Metodologi yang digunakan dalam Tugas akhir ini adalah sebagai berikut :

A. Studi literatur dan pemahaman sistem

Pada tahap ini dipelajari behan-bahan literatur Oracle 10g, pemrograman Java berorientasi obyek, dan algoritma *Support Vector Machine* (SVM).

B. Pengumpulan data

Pada tahap ini dilakukan pengumpulan data secara manual dari dokumen-dokumen elektronik berbahasa Inggris

C. Perancangan perangkat lunak

Pada tahap ini dilakukan perancangan terhadap perangkat lunak yang meliputi penentuan struktur data yang akan digunakan, proses – proses yang akan dilaksanakan, dan penentuan rancangan antar muka berdasarkan studi pustaka yang telah dilakukan.

D. Pembuatan perangkat lunak

Perangkat lunak akan diimplementasikan menggunakan Oracle 10g dengan konsep pemrograman Java berorientasi objek. Input yang akan diterima program berupa dokumen (*corpus*). Sedangkan outputnya akan berupa list-list klasifikasi dokumen hasil pengklasifikasian dokumen. Platform yang digunakan adalah Windows 2000 Professional atau Windows 2000 Advanced Server.

E. Pengujian dan evaluasi perangkat lunak

Pada tahap ini program yang telah dibuat diuji coba dengan *sample corpus* di atas. Selanjutnya hasil klasifikasi ini dievaluasi berdasarkan pengukuran tingkat kesalahan .

F. Penulisan Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan dan dokumentasi dari perangkat lunak, dari tahap awal hingga akhir.

1.6 SISTEMATIKA PENULISAN

Buku ini disusun dengan sistematika penulisan sebagai berikut:

BAB I Pendahuluan

Pendahuluan. Bab ini berisi latar belakang, permasalahan, tujuan, batasan masalah, metodologi dan sistematika pembahasan.

BAB II Information Retrieval.

Pada bab ini akan dibahas ilmu *information retrieval* yang mendukung pembuatan Tugas Akhir ini.

BAB III Oracle Text 10.1.0.2.0

Pada bab ini akan mengupas Oracle Text 10.1.0.2.0 dan fitur-fitur didalamnya.

BAB IV Perancangan dan Implementasi Perangkat Lunak.

Pada bab ini akan dibahas rancangan basisdata, rancangan proses serta rancangan antarmuka untuk perangkat lunak beserta pembuatan perangkat lunak di setiap bagian aplikasi.

BAB V Uji Coba Perangkat Lunak.

Pada bab ini akan dilakukan uji coba sesuai yang telah dijelaskan

pada metodologi tugas akhir ini.

BAB VI Penutup

Bab ini berisi kesimpulan yang dapat diambil dari Tugas Akhir ini beserta saran untuk pengembangan selanjutnya

BAB II

DASAR TEORI

BAB II
DASAR TEORI

Bab ini akan menjelaskan mengenai hal-hal yang menunjang penyusunan tugas akhir ini terutama mengenai information retrieval atau yang lebih dikenal dalam terminologi bahasa Indonesia sebagai temu kembali informasi.

2.1 INFORMATION RETRIEVAL

Sistem Temu Kembali Informasi (*Information Retrieval)* adalah suatu bidang ilmu yang mempelajari cara-cara menemukan kembali informasi dengan melakukan penelusuran dokumen atau kumpulan informasi yang telah di dapat dan disimpan sebelumnya.

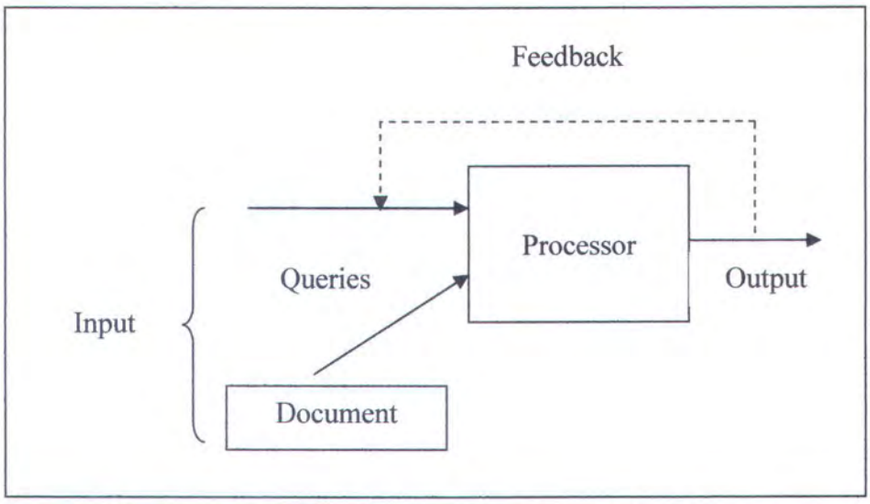
Diawali pada tahun 1940 dimana permasalahan penyimpanan informasi dan penelusuran kembali informasi menjadi perhatian yang cukup menarik, dengan ditemukannya permasalahan ketika informasi yang disimpan sudah mencapai jumlah yang cukup besar, proses penelusurannya akan menemui kesulitan dalam hal kecepatan akses dan keakuratan. Salah satu efek yang ditemui pada masa itu adalah, sering kali beberapa informasi yang relevan dan dibutuhkan diabaikan begitu saja, karena informasi tersebut tidak pernah atau sulit untuk terungkap, dan hal ini tentu saja mengakibatkan pekerjaan yang diulang kembali dengan tujuan untuk memperoleh informasi yang sama.

Secara prinsip, konsep penyimpanan dan pencarian kembali informasi adalah sederhana. Misalkan terdapat sekumpulan dokumen yang disimpan, dan kemudian ada pula kebutuhan untuk mendapatkan informasi tertentu dari

sekumpulan dokumen tersebut. Kebutuhan ini dapat diformulasikan sebagai sebuah pertanyaan (permintaan atau query) yang akan dijawab dengan sekumpulan dokumen yang mampu memuaskan kebutuhan tersebut akan informasi tertentu. Secara sederhana pula, pemenuhan kebutuhan tersebut dapat dicapai dengan menelusuri semua dokumen yang ada pada penyimpanannya, dan tentu saja hal ini akan sangat tidak efisien dari segi waktu dan tenaga sehingga hampir dipastikan tidak akan dapat dilakukan.

Diagram di bawah ini menunjukkan gambaran sederhana dari kinerja sebuah sistem temu kembali informasi, yang terdiri atas 3 elemen dasar yaitu : masukan (input), pemroses (processor) dan keluaran (output). Dimulai dari sisi masukan, permasalahan yang mendasar disini adalah bagaimana menentukan representasi dari setiap dokumen yang mampu di proses (query suitable) dalam sebuah komputer. Dengan penegasan bahwa dalam sebuah sistem temu kembali informasi berbasis komputer kebanyakan hanya menyimpan representasi dari dokumen (atau query) yang berarti bahwa isi dari dokumen akan hilang setelah diproses untuk mendapatkan bentuk representasinya. Sebuah representasi dari dokumen untuk sistem temu kembali informasi dapat berupa daftar kata yang signifikan. Dalam sistem ini juga lebih dipilih untuk menggunakan bahasa artifisial dari pada bahasa natural dalam proses komputer yang akan dijalankan untuk melakukan query dan memformulasikan dokumen. Tentu saja dengan asumsi bahwa pengguna dari sistem ini harus mengetahui bagaimana mengkespresikan kebutuhannya akan informasti tertentu dalam sebuah bentuk bahasa artifisial.

Bagian pemroses dalam sistem ini akan menjalankan proses temu kembali (retreival), proses ini akan biasanya terdiri atas : proses strukturisasi informasi dalam bentuk tertentu, misalkan dengan mengklasifikasikannya. Atau juga melibatkan proses nyata dari fungsi penemuan kembali, yaitu mengeksekusi strategi pencarian sebagai sebuah respon dari adanya query. Dalam diagram, dokumen diletakkan sebagai bagian yang terpisah untuk menegaskan bahwa dokumen - dokumen tersebut tidak hanya befungsi sebagai masukan, tetapi juga dapat dilibatkan selama proses penemuan kembali infomasi dijalankan.



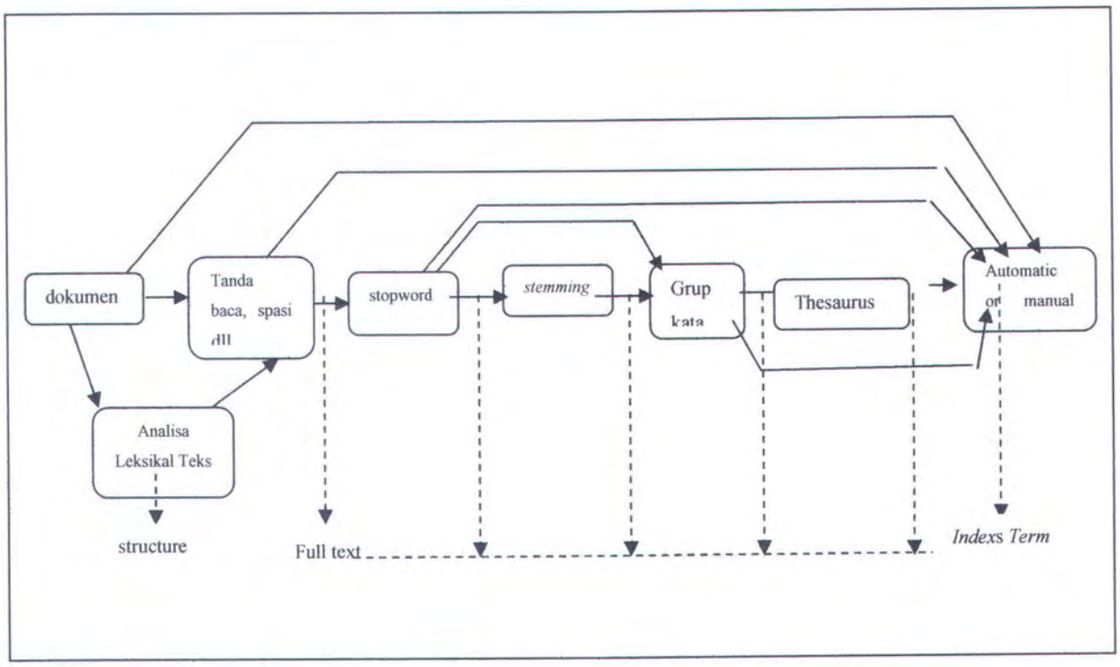
Gambar 2.1 Gambaran Umum Sistem Informasi Temu Kembali

Teknik IR banyak digunakan dalam tiga fase pengklasifikasian teks yang terdiri atas[Seb-2002]:

1. Persiapan dokumen (*tahap preprocessing*) termasuk proses *indexing.*
2. Pembuatan pengklasifikasi melalui proses pembelajaran, dalam tugas akhir ini menggunakan *decision tree classifier.*
3. Evaluasi unjuk kerja pengklasifikasi.

2.2 PREPROCESSING

Tahap ini merupakan tahap dimana dokumen tekstual ditangani dan disiapkan untuk membuat dokumen-dokumen tersebut siap diproses oleh *decision tree classifier*. Gambar 2.2 berikut ini merupakan bagan tahapan teks *preprocessing* yang dilakukan terhadap dokumen sebelum bisa dilakukan proses *training* didalamnya :



Gambar 2.2 Bagan Tahapan Teks Pre-processing

Ada 5 tahap prosedur teks *preprocessing* [Bae-1999] yaitu :

1. Analisa leksikal teks
2. Penghilangan *stopword*
3. *Stemming*
4. Pemilihan kata terindeks untuk menentukan grup kata
5. Kontruksi struktur kategorisasi kata (*thesaurus*)

2.2.1 ANALISA LEKSIKAL

Analisa leksikal(*lexycal analyzer*) adalah suatu proses transformasi teks yang masuk(dokumen) menjadi kata-kata(kata kandidat yang nantinya akan diadaptasikan sebagai indeks kata). Sekilas, tahap ini terlihat seperti pengidentifikasian spasi sebagai pemisah kata, tapi sebenarnya tahap ini lebih dari itu. Misalnya penanganan digit, tanda hubung, tanda baca, dan besar kecil huruf harus sangat diperhatikan. Ditambah lagi, nomer biasanya bukan indeks kata yang baik, karena tanpa memperhatikan isi kalimat, nomer bukanlah suatu kata yang berarti. Jadi biasanya nomer tidak dianggap dalam indeks kata. Tanda hubung juga merupakan keputusan yang sulit untuk *lexycal analyzer*. Misalnya *analyzer* ini memperlakukan '*state-of-the-art*' dan '*state of the art*' dengan cara yang sama. Padahal beberapa kata yang diberi tanda hubung, adalah satu bagian yang integral. Penetapan analisa leksikal ini akan dilakukan sesuai *preference* Lexer di Oracle Text 10.1.0.2.0.

2.2.2 PENGHILANGAN STOPWORD

Sejak awal perkembangan *information retrieval* telah dipelajari dan diamati kalau banyak jenis kata yang sering muncul dalam bahasa Inggris (seperti '*the*', '*of*', '*and*', '*to*', dll) tidak dapat digunakan sebagai *term* pengindeks. Kata-kata yang terlalu sering muncul ini disebut sebagai *stopword* dan biasanya dihilangkan dari list indeks kata yang potensial. Preposisi, kata hubung dan partikel biasanya merupakan kandidat list stopword yang disebut *stoplist*.

Suatu daftar kata yang akan dibuang dalam proses pengindeksan, karena mengurangi kualitas *term* pengindeks, disebut *stoplist* atau *negative dictionary* [Fox-1992].

Ada dua cara untuk melakukan penyaringan kata-kata *stoplist* dalam aliran kata, pertama, cek hasil keluaran dari *lexical analyzer* dan buang setiap *stopword* (kata-kata dalam *stoplist*), kedua, pembuangan *stopword* sebagai bagian dari analisis leksikal.

Untuk pendekatan pertama, penyaringan *stopword* dari keluaran *lexical analyzer*, akan memunculkan masalah terhadap usaha pencarian *stopword* dalam suatu tabel *stoplist*: karena setiap token harus dicari di dalam *stoplist*, kemudian baru bisa membuangnya.

Pendekatan kedua diyakini sebagai cara yang terbaik. Karena dengan mengikutsertakan pembuangan stopword pada proses analisa leksikal, pengenalan *stopword* dari daftar yang berukuran besar dapat dilakukan dengan *cost* lebih kecil. *Lexical analyzer* dapat dibuat secara otomatis, lebih mudah dan akan sedikit *error* yang mungkin muncul daripada dibuat secara manual. Cara ini pula yang digunakan pada *preference* Stoplist di proses *indexing* Oracle Text 10.1.0.2.0. *Stoplist* yang berisi kumpulan *stopword* akan diload dan otomatis dihilangkan pada proses *indexing*.

2.2.3 STEMMING

Dalam dokumen teks, suatu kata dapat muncul dalam berbagai bentuk variannya. Contoh, kata "order" juga dapat muncul dalam bentuknya yang lain seperti "ordering", "ordered", "orders", "orderly", dan sebagainya. Walaupun

kata-kata tersebut berbeda dalam bentuk tetapi mewakili konsep dasar yang sama. Dalam proses *information retrieval*, termasuk pengklasifikasi teks, sangat diharapkan untuk menggabungkan varian-varian dari kata yang sama tersebut ke dalam satu bentuk dasar. Dalam *information retrieval*, proses untuk menggabungkan atau memecahkan dari setiap varian-varian suatu kata menjadi kata dasar disebut *stemming* atau *conflation*. Hasil dari *stemming* disebut *stem* atau akar kata [Lam-1996]. *Stem* adalah bagian dari kata yang tersisa setelah dihilangkan imbuhannya(misalnya awalan dan akhiran). Sebagai contoh, kata *connect* yang merupakan *stem* dari berbagai kata variannya yaitu *connected*, *connecting*, *connection* dan *connections*.

Menurut Frakes [Fra-1992], teknik untuk meningkatkan performa dari *information retrieval* adalah dengan menyediakan jalan bagi para pencari data untuk menemukan varian dari *term* yang dicari. *Stemming* juga dapat dipergunakan untuk mengurangi ukuran dari file indeks. Algoritma *stemming* dapat dikategorikan secara garis besar kedalam algoritma-algoritma dibawah ini [Fra-1992] :

- *Successor Variety Stemmer*

  Dikembangkan oleh Hafer dan Weiss berdasarkan proses dalam struktur linguistik yang berusaha menentukan batas kata dan morfem berdasarkan distribusi fonem dalam suatu perkataan atau ungkapan yang jumlahnya besar. Metode *stemming* ini menggunakan huruf sebagai pengganti dari fonem, dan tubuh dari teks sebagai pengganti ungkapan yang tertulis secara fonem.

- *n-gram Stemmer*

Adamson dan Boreham melaporkan suatu metode untuk men-*conflate term* yang disebut metode *shared digram*. Suatu *digram* terdiri dari sepasang huruf yang susunannya berdampingan. Selain *digram* dapat pula mempergunakan bentuk *trigrams*, dan seterusnya, itu mengapa metode ini disebut n-*gram*. Walaupun ini disebut metode *stemming*, tetapi tidak menghasilkan akar kata.

- *Affix Removal Stemmer*

  Algoritma *affix removal* akan membuang sufiks dan prefiks dari *term* menjadi suatu *stem*. *Affix* (afiks) adalah imbuhan. Algoritma *affix removal* yang paling sering digunakan adalah *iterative longest match stemmer*, yang dikembangkan oleh Lovins. Algoritma ini akan membuang string terpanjang yang mungkin dari suatu kata berdasarkan himpunan aturan. Algoritma *iterative longest match stemmer* berikutnya dikembangkan oleh Salton, Dawson, Porter, Paice. Dari semua algoritma *affix removal stemmer*, yang paling sering dan umum digunakan adalah algoritma Porter karena esifisien dan kecepatannya sehingga algoritma ini pula yang digunakan dalam proses *indexing* Oracle Text 10.1.0.2.0 dalam tugas akhir ini.

2.2.3.1 ALGORITMA PORTER

Algoritma Stemmer Porter adalah sebuah algoritma *stemmer* yang diciptakan oleh Martin Porter dari Universitas Cambridge pada tahun 1980. Algoritma Porter (atau biasa disebut 'Porter *stemmer*') adalah suatu proses untuk menghilangkan akhiran morfologikal dan infleksional yang umum pada bahasa

Inggris. Algoritma ini sangat efektif dan sudah digunakan secara luas dan diterapkan dalam proses stemming pada Oracle Text 10.1.0.2.0.

Rule-rule pada algoritma Porter dibagi menjadi lima fase yang diaplikasikan pada kata-kata diteks dimulai dari fase 1 sampai fase 5. Selanjutnya, rule tersebut diaplikasikan secara sekuensial satu sama lain sebagai perintah dalam program [Bae-1999] Algoritma Porter dispesifikasikan dalam bahasa *pseudo programming* yang perintahnya diadaptasi ke konvensi semi-formal dibawah ini:

- Suatu variabel konsonan direpresentasikan dengan simbol *C* yang digunakan untuk menunjukkan semua huruf selain huruf *a,e,i,o,u* dan selain huruf *y*.
- Suatu variabel vokal direpresentasikan dengan simbol *V* yang digunakan untuk menunjukkan semua huruf selain konsonan.
- Suatu *generic letter* (bisa konsonan atau vokal) direpresentasikan dengan simbol *L*.
- Simbol 1#1 digunakan untuk menunjukkan *empty string* (misalnya yang tanpa huruf).
- Kombinasi dari *C*, *V*, and *L* digunakan untuk mendefinisikan *patterns*.
- Simbol * digunakan untuk menunjukan nol atau repetisi lebih dari yang diberikan oleh *pattern*.
- Simbol + digunakan untuk menunjukkan satu atau lebih repitisi dari *pattern* yang diberikan.
- Parenthesis yang sesuai digunakan untuk menempatkan variabel sekuen ke operator * dan +.

- *Generic pattern* merupakan kombinasi dari simbol, parenthesis yang sesuai, dan operator * dan +.
- Substitusi rule diperlakukan sebagai perintah yang dipisah dengan tanda baca semicolon.
- Rule subtitusi diaplikasikan ke sufiks pada kata.
- Pernyataan kondisional *if* dieks7presikan sebagai ``*if (pattern) rule*'' dan rule dieksekusi hanya jika kondisi *pattern* sesuai kondisi kata.
- Baris yang dimulai dengan a % diperlakukan sebagai komentar.
- *Curled bracket* digunakan untuk membentuk perintah gabungan.
- Suatu pernyataan ``*select rule with longest suffix*'' memilih rule tunggal untuk dieksekusi pada semua rule dalam perintah gabungan. Rule yang dipilih adalah yang paling sesuai dengan sufiks terpanjang.

Jadi ekspresi $(C)^*$ berarti sekuen dari nol atau lebih dari satu konsonan sedangkan ekspresi $((V)^*(C)^*)^*$ berarti sekuen dari nol atau lebih dari satu vokal yang diikuti oleh no atau lebih dari satu konsonan yang dimana bisa muncul nol atau lebih dari satu kali.

Misalnya perintah *if (*V*L) then ed* 2#21#1 menyatakan bahwa substitusi dari sufiks *ed* dengan nol(misalnya penghapusan sufiks *ed*) hanya terjadi jika kata tersebut berisi suatu vokal dan paling sedikit memiliki satu huruf tambahan.

Algoritma Porter diaplikasi ke tiap kata pada teks (formulasi sederhana) dan diberikan dengan prosedur sintaks berikut ini.

```
%Phase 1: Plurals and past participles.
select rule with longest suffix {

 sses 2#2 ss;
 ies 2#2 i;
```

```
 ss 2#2 ss;
 s 2#2 1#1; }

select rule with longest suffix {

 if ( (C)*((V)+(C)+)+(V)*eed) then eed 2#2 ee;
 if (*V*ed or *V*ing) then {

  select rule with longest suffix {

   ed 2#2 1#1;
   ing 2#2 1#1;

   }

  select rule with longest suffix {

   at 2#2 ate;
   bl 2#2 ble;
   iz 2#2 ize;
   if ((* C1C2) and ( C1 = C2) and ( 3#3 {l,s,z})) then C1C2 2#2 C1;
   if (( (C)*((V)+(C)+)C1V1C2) and ( 4#4 {w,x,y})) then C1V1C2 2#2

   C1V1C2e;

   }

 }

}

if (*V*y) then y 2#2 i;
if ( (C)*((V)+(C)+)+(V)*) then
 select rule with longest suffix {

  ational 2#2 ate;
  tional 2#2 tion;
  enci 2#2 ence;
  anci 2#2 ance;
  izer 2#2 ize;
  abli 2#2 able;
  alli 2#2 al;
  entli 2#2 ent;
  eli 2#2 e;
  ousli 2#2 ous;
  ization 2#2 ize;
  ation 2#2 ate;
  ator 2#2 ate;
  alism 2#2 al;
  iveness 2#2 ive;
  fulness 2#2 ful;
  ousness 2#2 ous;
  aliti 2#2 al;
  iviti 2#2 ive;
  biliti 2#2 ble;

 }

if ( (C)*((V)+(C)+)+(V)*) then
```

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH NOPEMBER

```
 select rule with longest suffix {

  icate 2#2 ic;
  ative 2#2 1#1;
  alize 2#2 al;
  iciti 2#2 ic;
  ical 2#2 ic;
  ful 2#2 1#1;
  ness 2#2 1#1;

 }

if ( (C)*((V)+(C)+)((V)+(C)+)+(V)*) then
 select rule with longest suffix {

  al 2#2 1#1;
  ance 2#2 1#1;
  ence 2#2 1#1;
  er 2#2 1#1;
  ic 2#2 1#1;
  able 2#2 1#1;
  ible 2#2 1#1;
  ant 2#2 1#1;
  ement 2#2 1#1;
  ment 2#2 1#1;
  ent 2#2 1#1;
  ou 2#2 1#1;
  ism 2#2 1#1;
  ate 2#2 1#1;
  iti 2#2 1#1;
  ous 2#2 1#1;
  ive 2#2 1#1;
  ize 2#2 1#1;
  if (*s or *t) then ion 2#2 1#1;

 }

select rule with longest suffix {

 if ( (C)*((V)+(C)+)((V)+(C)+)+(V)*) then e 2#2 1#1;
 if (( (C)*((V)+(C)+)(V)*) and not (( *C1V1C2) and ( 4#4 {w,x,y})))

 then e 2#2 nil;

 }

if ( (C)*((V)+(C)+)((V)+(C)+)+V*ll) then ll 2#2 l;
```

Langkah-langkah prosedural diatas diaplikasikan pada kata-kata dalam teks mulai dari langkah 1 dan bergerak sampai ke langkah 5.

- Langkah 1 : berhubungan dengan akhiran bentuk jamak dan bentuk lampau.

  Misalnya: *plastered* → *plaster*, *motoring* → *motor*

- Langkah 2 : berhubungan dengan kecocokan pola pada beberapa akhiran yang umum. Misalnya: *happy* → *happi*, *relational* → *relate*, *callousness* → *callous*
- Langkah 3 : berhubungan dengan akhiran kata yang khusus.

  Misalnya: *triplicate* → *triplic*, *hopeful* → *hope*
- Langkah 4 : mengecek potongan kata pada akhiran yang lebih panjang untuk menangani kata gabungan.

  Misalnya: *revival* → *reviv*, *allowance*→*allow*, *inference*→*infer* dan lain sebagainya.
- Langkah 5 : mengecek jika potongan kata berakhiran karakter vokal dan memperbaikinya supaya sesuai.

  Misalnya : *probate*→*probat*,*cease*→*ceas*, *controll*→*control*.

2.2.4 PEMILIHAN KATA TERINDEKS UNTUK MENENTUKAN GRUP KATA

Suatu representasi teks penuh tidak bisa begitu saja langsung dianggap sebagai kata terindeks (kata-kata hasil proses preprocessing) karena bisa mengakibatkan naiknya waktu komputasi. Karena itu bisa diadaptasikan suatu *abstract view* dimana tidak semua kata digunakan sebagai kata terindeks. Pendekatan yang baik adalah dengan pengindentifikasian grup kata benda(sebagaimana yang dilakukan pada Inquery system).

Kalimat pada umumnya terdiri dari gabungan kata benda, kata ganti, partikel, kata kerja, kata sifat, kata keterangan dan kata penghubung. Saat kata-kata dari tiap kelas grup kata tersebut digunakan untuk tujuan tertentu, dapat dipastikan arti terbesar dibawa oleh kata benda. Sehingga strategi termudah untuk pemilihan kata terindeks otomatis adalah dengan menggunakan kata benda pada

teks. Hal ini dapat dilakukan melalui penghilangan kata kerja, kata sifat, kata keterangan, kata penghubung, partikel dan kata ganti secara sistematis.

Proses menggabungkan dua atau tiga kata menjadi komponen tunggal(misalnya *computer science*) adalah hal yang umum, sehingga hal ini mengharuskan pengkelasan kata benda yang muncul diteks menjadi komponen pengindeksan tunggal. Hal itu bisa dilakukan dengan mengadaptasikan *noun group*(grup kata benda). *Noun group* adalah seperangkat kata benda yang jarak sintaktisnya pada teks(dihitung pada jumlah kata-kata antara 2 kata benda) tidak melebihi *threshold*(batasan) yang ditentukan(misalnya 3). Misalnya pada kelompok kata *reverse engineering software development*, kelompok kata ini melebihi jumlah *threshold* (yaitu 3). Sehingga kata bisa dipecah menjadi 2 menjadi *reverse engineering* dan *software development*. Saat *noun group* diadaptasikan sebagai kata terindeks, akan didapatkan pandangan logikal konseptual pada seperangkat kata yang bukan kata terindeks dasar.

Proses pemilihan kata terindeks ini tidak dilakukan oleh Oracle Text 10.1.0.2.0 karena mengingat sulitnya menentukan penggabungan kata dengan memperhitungkan identifikasi kelompok kata. Apalagi pada bagan preprocessing tahap grup kata ini boleh dilompati.

2.2.5 KONTRUKSI STRUKTUR KATEGORISASI KATA (*THESAURUS*)

Kata *Thesaurus* berasal dari bahasa Yunani dan Latin dan digunakan sebagai referensi perbendaharaan kata-kata atau istilah pada bidang pengetahuan tertentu. Pada bentuk yang sederhana, perbendaharaan kata-kata ini terdiri dari :

1. List kata-kata penting yang diberikan berdasarkan bidang pengetahuannya

2. Untuk tiap kata pada list ini, ada seperangkat kata yang berhubungan. Kata yang berhubungan tersebut diambil dari hubungan sinonimitas dengan berbagai variasinya.

Pada Sub bab berikutnya akan lebih dijelaskan tentang konstruksi model didalam thesaurus.

2.2.5.1 THESAURUS INDEKS KATA

Biasanya suatu kata pada thesaurus digunakan untuk menunjukkan suatu konsep yang berdasarkan unit semantik untuk menyampaikan ide. Kata-kata tersebut dapat berupa kata tunggal, grup kata atau frase, tapi sebagaian besar merupakan kata tunggal. Kata-kata tersebut biasanya berupa kata benda karena kata benda adalah bagian terkonkret dari suatu kalimat. Kata-kata tersebut juga bisa berupa kata kerja dalam bentuk *gerund* sehingga bisa digunakan sebagai kata benda (misalnya : *acting*, *teaching*, dsb).

Saat suatu konsep tidak dapat diekpresikan dengan kata tunggal, grup kata bisa digunakan sebagai gantinya. Misalnya, banyak konsep diekspresikan lebih baik dengan kombinasi kata sifat dan kata benda. Contoh umum adalah *ballictic missiles*. Pada kasus ini, pengindeksan kata gabungan ini akan langsung menghasilkan sebuah entri dibawah kata *ballistic* dan tidak ada entri dibawah kata *missiles*. Untuk menghindari masalah ini, kata gabungan biasanya dimodifikasi dengan menulis kata benda sebagai kata pertama. Misalnya, dengan mengganti kata gabungan menjadi *missiles*, *ballistic*.

Penggunaan kata jamak *missiles* lebih disukai dari penggunaan kata tunggal, *missile*. Alasannya karena thesaurus menampilkan kelas dari benda-benda dan

biasanya berupa bentuk jamak. Bagaimanapun, bentuk tunggal digunakan untuk kata gabungan yang biasanya berbentuk tunggal seperti *body temperature*. Menentukan antara penggunaan kata tunggal dan jamak bukanlah suatu hal yang sederhana.

Selain kata tersebut, biasanya perlu untuk menambah entri thesarurus dengan definisi atau penjelasan. Alasannya karena perlu untuk menspesifikasikan arti kata yang lebih akurat pada konteks dari thesaurus tertentu. Sebagai contoh, kata *seal* punya arti pada konteks *marine animals* dan punya arti yang khusus pada konteks *documents*. Pada kasus tersebut, definisi harus didahului dengan penjelasan konteks seperti *seal(marine animals)* dan *seal(documents)*.

2.2.5.2 THESAURUS KATA BERHUBUNGAN

Seperangkat kata yang berhubungan pada kata thesaurus biasanya dibentuk dari sinonim dan sinonim yang dekat. Ditambah lagi, suatu hubungan dapat diinduksi dari pola kejadian antara dokumen-dokumen. Hubungan tersebut pada umumnya berhirarki dan bisanya diindikasikan dengan kata berhubungan yang *broader* (meluas) direpresentasikan dengan BT atau *narrower* (menyempit) direpresentasikan dengan NT. Bisa dikatakan bahwa kata-kata tersebut saling *related* (berhubungan) direpresentasikan dengan RT.

Hubungan BT dan NT didefinisikan sebagai hirarki klasifikasi dimana *broader term* yang berhubungan dengan suatu kelas akan berhubungan juga dengan *narrower term* pada kelasnya. Lebih detilnya, suatu *narrower term* bisa berhubungan dengan 2 atau lebih *broader terms* (yang walaupun bukan merupakan suatu kasus yang sering terjadi). Walaupun hubungan BT dan NT bisa

diidentifikasikan dengan cara otomatis penuh (tanpa bantuan manusia), tapi menangani hubungan RT akan lebih sulit. Salah satunya alasannya karena hubungan RT lebih bergantung pada konteks spesifiknya dan kebutuhan tertentu dari kelompok pengguna dan hal itu lebih sulit untuk diidentifikasikan tanpa pengetahuan dari spesialis. Bentuk konstruksi *thesaurus* berdasarkan hubungan kata inilah yang diterapkan dalam *thesaurus* Oracle Text 10.1.0.2.0 dan akan dipakai dalam tugas akhir ini.

2.3 PENGINDEKSAN(*INDEXING*)

Suatu indeks adalah bahasa yang digunakan untuk merepresentasikan dokumen dan permintaan pengguna. Elemen dari bahasa indeks adalah istilah (*keyword*, *term*, *feature*) pengindeksan, yang dapat diperoleh dari teks suatu dokumen yang akan digambarkan, atau bisa juga diperoleh secara independen.

Menurut Zobel dan Moffat [Zob-1998], suatu *term* adalah konsep yang teridentifikasi di dalam suatu dokumen. Untuk dokumen teks, setiap kata di dalam dokumen adalah *term*, setelah kata tersebut mengalami proses pembentukan kata dasar (*stemming*) dan transformasi sejenisnya. Tetapi suatu *term* mungkin saja berbentuk pasangan kata, suatu frase, atau suatu deskripsi yang tidak muncul di dalam dokumen tersebut.

Kosakata suatu bahasa indeks dapat bersifat terkontrol atau tidak terkontrol. Kosakata terkontrol yaitu kosakata yang berasal dari suatu daftar yang telah disusun untuk digunakan oleh pengindeks.

Menurut Sebastiani [Seb-2002], pengindeksan diperlukan karena teks tidak dapat dimengerti secara langsung oleh pengklasifikasi. Oleh karena itu, prosedur pengindeksan, yang memetakan dokumen teks ke dalam suatu representasi dengan menggunakan teks yang terkandung di dalamnya, harus diterapkan pada dokumen pelatihan, validasi, dan pengujian. Suatu teks di dalam dokumen teks akan direpresentasikan ke dalam vektor bobot *term* $[d_j] = \langle w_{1j}, \ldots, w_{|T|j} \rangle$, di mana T adalah himpunan istilah (yang terkadang disebut sebagai *term/feature*, di dalam tulisan ini akan mempergunakan frase "himpunan *term*") yang muncul minimal sekali pada minimal satu dokumen dari koleksi dokumen dalam himpunan dokumen pelatihan (T_r), dan $0 \leq w_{kj} \leq 1$ mewakili seberapa besar suatu *term* t_k, t_k adalah *term* yang ke-k, berperan dalam memberikan informasi dokumen d_j, d_j adalah dokumen yang ke-j. Himpunan *term* tersebut dipergunakan untuk mengindeks dokumen teks atau merepresentasikan setiap dokumen teks.

Dalam menghitung bobot yang merupakan representasi dari *term* dalam dokumen terdapat beberapa pendekatan yang tergantung dari:

a. Beragam pengertian terhadap istilah *term*;
b. Beragam cara untuk menghitung bobot suatu *term*.

Bentuk pilihan untuk a adalah dengan mengidentifikasikan *term* dengan kata-kata. Hal ini seringkali disebut dengan pendekatan *set of words* atau *bag of words* terhadap representasi dokumen, tergantung dari bobotnya biner atau bukan. Maksudnya yaitu dengan menyediakan terlebih dahulu kumpulan kata-kata penting (biasanya berupa kata benda) sebagai kontroller dari *term*, sehingga hanya kata-kata yang ada pada kontroller tersebut yang dibobotkan.

Untuk b, bobot biasanya berkisar antara 0 dan 1. Pada hal-hal tertentu, bobot biner dapat digunakan (1 menyatakan kehadiran, 0 menyatakan ketidakhadiran suatu *term* di dalam dokumen). Karena dalam tugas akhir ini digunakan *decision tree classifier* maka digunakan pengindeksan nonbiner dalam menentukan bobot w_{kj} untuk suatu *term* t_k di dokumen d_j untuk merepresentasikan dokumen dalam bentuk vektor bobot *term*. Dalam Oracle Text, pembobotan menggunakan fungsi standar *tfidf*, yang diperkenalkan oleh Salton dan Buckley, dengan persamaan:

$$tfidf(t_k, d_j) = \#(t_k, d_j) \bullet \log \frac{|T_r|}{\#_{Tr}(t_k)} \tag{2.1}$$

di mana $\#(t_k, d_j)$ menyatakan jumlah kemunculan *term* t_k di dalam dokumen d_j, dan $\#_{Tr}(t_k)$ menyatakan frekuensi dokumen untuk *term* t_k, yaitu, jumlah dokumen di dalam himpunan dokumen pelatihan T_r di mana *term* t_k muncul. Fungsi tersebut didasarkan pada intuisi-intuisi, pertama, semakin sering suatu *term* muncul di dalam dokumen, maka semakin tinggi tingkat representasinya untuk dokumen tersebut, dan, kedua, semakin banyak dokumen yang mengandung suatu *term*, maka semakin rendah tingkat diskriminan dari *term* tersebut.

Supaya bobot setiap *term* jatuh pada kisaran [0,1], dan supaya dokumen dapat direpresentasikan dengan vektor yang panjangnya sama, maka bobot-bobot hasil dari *tfidf* harus dinormalisasikan dengan fungsi *cosine normalization*, yaitu:

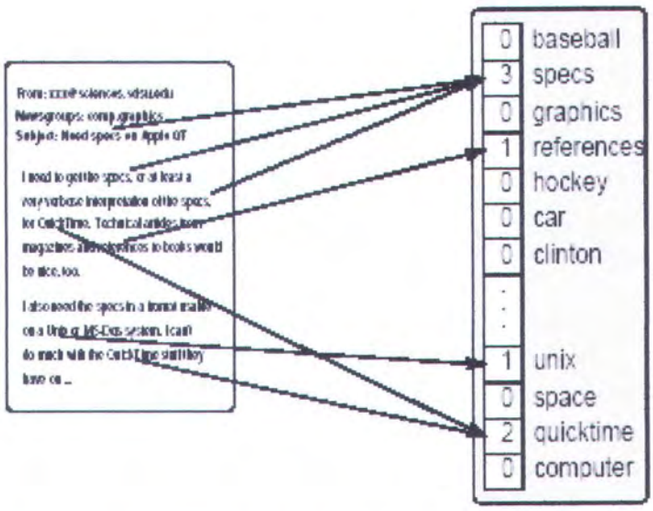
$$w_{kj} = \frac{tfidf(t_k, d_j)}{\sqrt{\sum_{s=1}^{|T|} (tfidf(t_s, d_j))^2}} \tag{2.2}$$

2.4 *MACHINE LEARNING* DALAM PENGKLASIFIKASIAN DOKUMEN

Tujuan dari proses pengklasifikasian teks atau dokumen adalah untuk mengelompokkan serangkaian dokumen kedalam satu kelompok yang telah didefinisikan sebelumnya. Setiap dokumen dapat temasuk kedalam beberapa kelompok, satu kelompok saja, atau tidak termasuk ke dalam kelompok manapun. Dengan menggunakan machine learning, dengan tujuan untuk melatih sistem pengklasifikasi, yang nantinya akan dapat mengenali kelas - kelas dari dokumen secara otomatis. Metode ini tentu saja teramasuk kedalam salah satu cabang machine learning yaitu metode supervised learning. Karena dimungkinkan terjadinya overlapping diantara kategori - kategori dokumen, maka setiap kategori akan diperlakukan sebagai permasalahan klasifikasi biner, dimana nantinya untuk setiap kategori, akan diperiksa apakah suatu dokumen masuk ke dalamnya atau tidak.

Langkah pertama yang harus diambil dalam pengklasifikasian teks adalah mentransformasi dokumen (yang biasanya merupakan sekumpulan dari karakter), menjadi sebuah representasi yang cocok untuk algoritma pembelajaran dan proses klasifikasi. Para pakar Information Retreival menyarankan bentuk kata dasar merupakan representasi yang sesuai untuk permasalahan ini. Urutan dari kata – kata ini pun dapat diabaikan Setiap kata akan dihubungkan dengan sebuah fitur, yang merupakan jumlah kemunculan kata tersebut dalam dokumen. Dan fitur - fitur kata ini akan ditampung ke dalam sebuah vektor. Untuk menghindari terbentuknya vektor yang terlalu besar maka para ilmuwan menyarankan agar kata yang di jadikan sebagai fitur, hanya kata - kata yang kemunculannya lebih dari

suatu batas nilai tertentu, misalnya 3 dan kata - kata tersebut bukan termasuk ke dalam stop-word (seperti kata dalam bahasa inggris 'like', 'and', dan sebagainya).



Gambar 2.3 Merepresentasikan teks dokumen sebagai vector

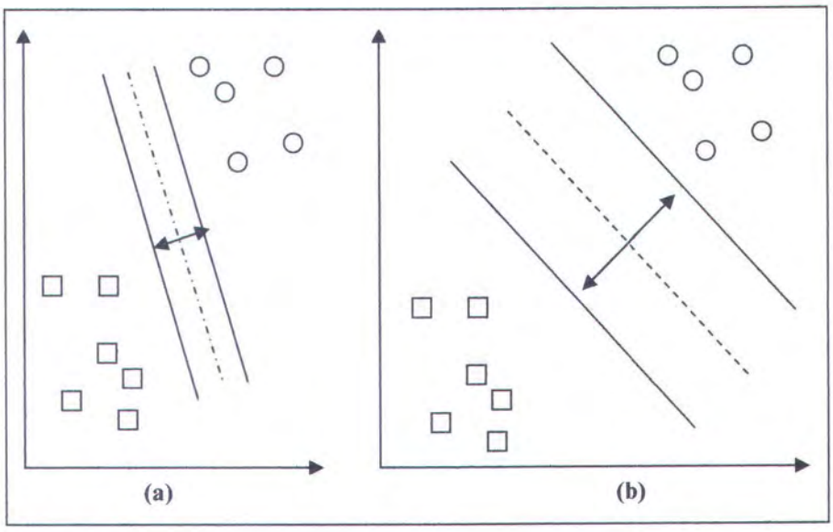
Kebanyakan pendekatan pembelajaran mesin untuk pengkategorian teks berbeda dalam pelaksanaan tahap pembuatan pengklasifikasi melalui proses induktif. Adapun metode-metode pembelajaran yang biasanya digunakan pada machine learning antara lain: *decision tree classifier, naive bayes classifier*, dan *nearest neighbour.* Sedangkan pada tugas akhir ini akan dititikberatkan pada penggunaan metode *support vector machine classifier* untuk pembelajarannya karena model yang dihasilkan diharapkan akan lebih akurat.

2.4.1 SVM CLASSIFIER

Tujuan dari metode *support vector machine* adalah untuk menemukan hyperplane optimal yang akan mengklasifikasikan data dengan benar dengan cara memisahkan nilai – nilai diantara dua kelas sejauh mungkin. Jika terdapat data set

yang terdistribusikan diantara 2 kelas, SVM akan berfungsi untuk mencari hyperplane yang akan :

- Menghasilkan pecahan terbesar yang mungkin ada diantara nilai - nilai yang berada pada kelas yang sama dan pada sisi yang sama dan memaksimalkan jarak diantara dua kelas
- Pencarian hyperplane yang optimal ini harus dapat meminimalkan kesalahan klasifikasi terhadap data training dan data testing yang belum pernah di uji coba sebelumnya. Pembahasan ini kemudian mengarah kepada teorema *Structural Risk Minimization.*



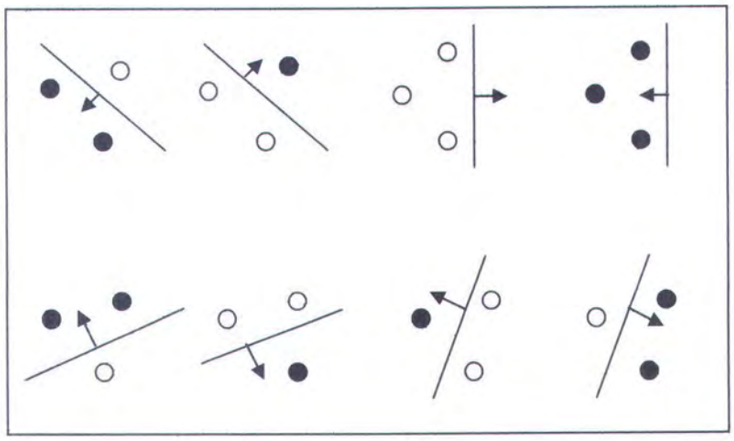
Gambar 2.4 (a) Hyperplane pemisah dengan margin kecil. (b) Hyperplane pemisah dengan margin besar

Support Vector Machine adalah metode yang dibuat berdasarkan prinsip Structural Risk Minimization (SRM) [Vap -1995], prinsip ini dikenal dalam teori - teori pembelajaran komputasi. Ide dasar dari SRM adalah bagaimana kita dapat

menemukan nilai hypothesis h yang dapat menjamin nilai kesalahan (true error) terkecil. True error dari hypothesis adalah merupakan kemungkinan bahwa nilai h akan menghasilkan kesalahan pada saat dilakukan proses pada data test (yang belum pernah dikenali sebelumnya).

2.4.2 VC DIMENSION

VC (Vapnik-Chervonenkis) Dimension adalah properti dari serangkaian fungsi $\{\boldsymbol{f}(\alpha)\}$, sehingga $\boldsymbol{f}(\mathbf{x}, \alpha) \in \{-1,1\}\ \forall\ \mathbf{x},\alpha$. Jika terdapat serangkaian nilai sebanyak $\boldsymbol{l}$ yang dapat di beri label kelas dengan $\mathbf{2}^{\boldsymbol{l}}$ kemungkinan. Dan untuk setiap kali pelabelan, apabila nilai - nilai anggota dalam himpunan $\{\boldsymbol{f}(\alpha)\}$ dapat dihitung dengan benar sehingga dapat dengan benar juga menghubungkan label-label, maka VC dimension untuk serangkaian fungsi $\{\boldsymbol{f}(\alpha)\}$ dapat didefinisikan sebagai jumlah nilai training maksimal yang dapat dihasilkan oleh $\{\boldsymbol{f}(\alpha)\}$. Fungsi $\boldsymbol{f}(\alpha)$ biasanya disebut sebagai *hypothesis*, dan himpunan $\{\boldsymbol{f}(\alpha) : \alpha \in \Lambda\}$ disebut sebagai *hypothesis space* dan di simbolkan dengan $\mathcal{H}$. *VC dimension* untuk sekumpulan fungsi $\{\boldsymbol{f}(\alpha)\}$ dalam hal ini adalah $\mathcal{H}$ didefinisikan sebagai jumlah nilai trainning maksimum yang bisa dipecahkan oleh $\mathcal{H}$ [Bur-1998].



Gambar 2.5 Kombinasi VC Dimension untuk R^2

Misalkan ruang dimana data berada adalah dalam $\mathcal{R}^2$, dan sekumpulan fungsi $\{f(\alpha)\}$ terdiri atas garis lurus berarah / *oriented lines*. Sehingga setiap garis akan membagi nilai – nilai yang ada kedalam tepat dua kelas, kelas 1 dan kelas -1. Orientasi dalam gambar 2.5 ditunjukkan dengan tanda panah, yang menjelaskan sisi mana dari garis yang merupakan kelas dengan label 1. Selama dimungkinkan untuk menemukan 3 nilai yang dapat dipecah dengan sekumpulan garis (oriented line) ini, maka tidak mungkin untuk menemukan pemecahan terhadap 4 titik. Sehingga VC dimension dari *set of oriented lines* dalam $\mathcal{R}^2$ adalah 3. Dan VC dimension untuk oriented hyperplane dalam $\mathcal{R}^n$ adalah n + 1.

2.4.3 Support Vector Machine

Algoritma SVM berdasar kepada struktur beberapa *hyperplane – hyperplane* pemisah, Jika terdapat hyperplane dengan parameter w dan b, dimana (w,b) $\in$ *S* x **R**,

$$\{x \in S : (w.x) + b = 0\} \tag{2.3}$$

Dengan :

$$\min_{i=1,\ldots,n} |(w.x_i) + b| = 1 \tag{2.4}$$

Jika w dan b di skala sehingga menjadi nilai yang terdekat dengan hyperplanne, dengan *distance* $\frac{1}{\|w\|}$. Sehingga margin diantara dua kelas, diukur dengan garis tegak lurus ke hyperplane dengan jarak minimal $\frac{2}{\|w\|}$. Maka algoritma SVM bertujuan untuk memaksimalkan margin $\frac{1}{\|w\|}$, dengan meminimalkan ||w||, dan mencari *VC dimension* terkecil yang sesuai.

Jika terdapat sekumpulan nilai $x_i \in \mathcal{R}^n$ dengan i = 1, 2, ... N. Dan setiap nilai x_i termasuk kedalam salah satu kelas dengan label $y_i \in \{-1,1\}$. Maka *set* S akan secara linear terpisah jika terdapat nilai $w \in \mathcal{R}^n$ dan $b \in \mathcal{R}^n$ sedemikian hingga :

$$y_i(w.x_i + b) \geq 1, i = 1,2,\ldots,N. \tag{2.5}$$

Pasangan (w,b) akan membentuk *hyperplane* dengan rumus $w \cdot x_i + b = 0$, dan disebut sebagai hyperplane pemisah. Jarak bertanda d_i dari nilai x_i dari hyperplane pemisah dihitung dengan rumus :

$$d_i = \frac{w.x_i + b}{\| w \|} \tag{2.6}$$

Dengan persamaan 2.5 dan 2.6, untuk semua $x_i \in S$, akan didapatkan :

$$y_i d_i \geq \frac{1}{\| w \|}$$

$$\forall x_i \in S \quad y_i d_i \geq \frac{1}{\| w \|} \tag{2.7}$$

Oleh sebab itu, $\frac{1}{\| w \|}$ adalah batas terendah (lower bound) pada jarak antara nilai x_i dengan separating *hyperplane* (w,b). *Separating hyperplane* yang optimal adalah *hyperplane* dimana jarak dari nilai terdekat dari S adalah maksimum.

2.4.4 KEUNGGULAN SVM UNTUK PENGKLASIFIKASIAN TEKS.

Metode apa saja yang cocok untuk permasalahan klasifikasi teks, adalah metode yang sesuai dengan karakteristik / *property* dari teks itu sendiri [Joa-1998]. Karakteristik tersebut antara lain :

1. *High dimensional input space* : data teks yang akan diklasifikasikan, harus diolah manjadi *feature* yang sesuai dengan model *classifiers*. Proses ini akan menghasilkan sangat banyak fitur (dapat melebihi 10000 fitur), karena SVM menggunakan pendekatan *overfitting protection*, sehingga

performanya secara keseluruhan tidak tergantung kepada banyaknya fitur, dan sangat potensial untuk menangani permasalahan ini.

2. *Few irrelevant features* : Salah satu cara mengatasi banyaknya fiturm, adalah dengan mengasumsikan akan banyak pula fitur – fitur yang tidak relevan. Sehingga muncul metode pemilihan fitur yang akan menghilangkan fitur – fitur yang tidak relevan ini. Namun dalam permasalahan klasifikasi teks, fitur yang tidak relevan ini sangat sedikit sekali di temui. Pada percobaannya Joachim [Joa-1998] berhasil menunjukkan bahwa, meskipun beberapa fitur dinyatakan tidak relevan, namun ternyata masih memiliki infromasi ynag layak untuk diperhatikan sehingga tetap relevan. Classifier yang tetap menggunakan fitur – fitur tersebut masih menunjukkan performa yang lebih baik. Hal ini menunjukkan bahwa sebuah classifier yang baik seharusnya tetap mengkombinasikan banyak fitur, daripada harus kehilangan informasi karena pemilihan fitur yang terlalu agresif.
3. Sebagian besar aplikasi klasifikasi teks adalah terpisah secara *linear* : Ide dasar dari SVM adalah mencari pemisah – pemisah linear (atau polynomial, RBF, dan sebagainya) tersebut.

Argumen – argumen diatas, memberikan bukti teoritis, bahwa SVM mampu menunjukkan performa yang baik untuk klasifikasi teks.

BAB III

ORACLE TEXT 10.1

BAB III
ORACLE TEXT 10.1

Pada bab ini akan dibahas mengenai Oracle Text 10.1. Termasuk didalamnya penjelasan mengenai fitur – fitur yang terdapat Oracle Text release 10 ini.

3.1 GAMBARAN UMUM ORACLE TEXT

Oracle Text merupakan teknologi yang pembangunan sebuah aplikasi *text query* dan aplikasi pengklasifikasian dokumen. Oracle Text menyediakan fasilitas *indexing*, pencarian kata dan tema, serta memiliki kemampuan untuk menampilkan teks.

Oracle Text 10.1.0.2 merupakan bagian dari standart Oracle10g dan Enterprise Editions yang menambahkan fitur pencarian teks yang powerful dan manajemen teks yang intellegent pada Oracle10g yang sekaligus mampu mengekstraksi berbagai format dokumen, melakukan *preprocessing* (tahap persiapan dokumen agar bisa dilakukan proses didalamnya) dan berbagai proses teks lainnya. Kemampuan dan kehandalannya yang sudah teruji dalam berbagai riset dan konferensi teknologi informasi tingkat dunia membuat aplikasi pengklasifikasian dokumen multiformat ini bisa diterapkan diatasnya.

Kelebihan yang dimiliki oracle text antara lain :

- sudah terintegrasi penuh dengan Oracle 10g
- menawarkan beberapa kelebihan dalam proses manajemen teks, servis berbasis dokumen dan XML.

- dapat melakukan pencarian teks dalam beberapa bahasa

Dengan telah diintegrasikannya oracle text sebagai salah satu fitur dalam database sistem, maka hal ini juga akan menawarkan keunggulan dalam keamanan, performa, skalabilitas, dan pengembangan aplikasi universal secara virtual.

Dan pengintegrasian kemampuan pencarian teks ke dalam fungsi SQL tunggal, pencarian informasi secara fleksibel, dan handal dengan berbagai macam koleksi dokumen akan sangat mudah untuk dilakukan.

Application Programming Interface (API) Oracle Text SQL juga sangat memudahkan bagi para pembangun aplikasi untuk menjalankan pengindeks-an teks dan menjalankan perintah pencarian teks.

Tipe indeks yang digunakan untuk membangun aplikasi klasifikasi adalah tipe indeks `CTXRULE`. Untuk membuat indeks `CTXRULE` butuh tabel yang berisi query-query yang juga bisa dibuat sendiri untuk kemudian diindeks. Dengan indeks `CTXRULE`, operator `MATCHES` bisa digunakan untuk mengklafikasikan dokumen tunggal. Sayangnya indeks `MATCHES` hanya bisa digunakan pada dokumen berformat plain text, HTML dan XML, sehingga tidak bisa digunakan untuk aplikasi tugas akhir ini. Namun demikian tetap akan dijelaskan pada subbab berikutnya ini karena merupakan aplikasi routing pengklasifikasi murni dari Oracle Text.

3.2 PERBEDAAN ORACLE TEXT 9.2 DENGAN ORACLE TEXT 10.1

Dalam tugas akhir ini sebagian besar pemrosesan dokumen dan klasifikasi akan dilakukan dengan Oracle Text 10.1, pada sub bab ini akan dijelaskan

perbedaan yang mendasar antara Oracle Text 9.2 dengan Oracle Text 10.1. Perbedaan yang akan dibahas disini menyangkut metode pelatihan atau pembelajaran dalam proses klasifikasi yang disediakan dalam oracle text 9.2 dan oracle text 10.1 dan lebih khusus lagi di dalam kelompok metode *supervised learning*. Pembahasan ini ditambahkan dengan tujuan untuk mendukung uji coba yang akan dilakukan dalam pembuatan tugas akhir, dimana akan dilakukan perbandingan hasil uji coba antara *Decision Tree Classifier* yang dibuat oleh nuning [Pur-2004], dengan *SVM Classifier*.

3.2.1 *DECISION TREES CLASSIFIER* DALAM ORACLE TEXT 9.2

Decision Tree secara umum didefinisikan sebagai sebuah metode yang akan menentukan pilihan diantara dua atau lebih pilihan yang tersedia. Dalam kasus klasifikasi dokumen, pemilihan ini diarahkan kepada dokumen mana yang cocok (matces) dengan data trainning atau dokumen mana saja yang tidak cocok. Sebuah model *decision tree* memiliki sekumpulan atribut yanh akan dites. Untuk kasus ini, atribut – atribut terserbut dapat berupa kata – kata dari dokumen, kata dasar – kata dasar dari dokumen dan tema dari dokumen.

Algoritma pembelajaran dalam Oracle Text, khususnya Oracle Text 9.2 akan membangun satu atau lebih *decision tree* untuk setiap kategori yang terdapat pada data *trainning*. *Decision tree* ini nantinya akan dikodekan menjadi sekumpulan *query* yang dapat digunakan oleh indeks CTXRULE.

Dalam pembelajaran *decision tree* dilibatkan konsep nilai *confidence*. Setiap aturan / *rule* yang dihasilkan akan diberikan sebuah nilai presentase dari akurasi dari *rule* terhadap data *trainning*. Dalam contoh yang sederhana, nilai akurasi ini

hampir selalu mencapai 100 %, namun hal ini tentu saja merupakan reprentasi dari keterbatasan dari data trainning.

Metode pembelajaran decision tree adalah sebuah metode untuk memperkirakan nilai diskrit dari fungsi target, dimana fungsi pembelajar akan direpresentasikan oleh decision tree [mitc-1997]. Decision tree akan mengklasifikasikan data dengan mengurutkannya ke dalam struktur pohon mulai dari akar sampai ke beberapa *leaf node*, yang menyediakan mekanisme klasifikasi data. Setiap node dari tree menentapkan tes untuk setiap atribut klasifikasi, dan setiap cabang yang muncul dari node berhubungan dengan setiap nilai yang mungkin dihasilkan dari setiap atribut. Data diklasifikasikan dengan diawali dari node root dari tree, dilakukan pengetesan atas atribut pada node tersebut, kemudian bergerak turun ke percabangannya, berdasarkan nilai dari atribut yang dimasukkan sebagai contoh. Proses ini akan diulang untuk sub tree yang berawal dari node baru (setelah percabangan). Kebanyakan algoritma yang dibangun untuk pembelajaran *decision tree*, merupakan variasi dari algoritma dasar yang merupakan struktur *top-down* dan pencarian secara menyeluruh / *greedy* melalui ruang yang mungkin ada dalam decision tree Permasalahan utamanya adalah pada pemilihan atribut mana yang akan di tes pada setiap node dalam tree. Hal ini ditentukan berdasarkan hasil pengukuran, semacam pencarian informasi, untuk mengevaluasi sebaik apa sebuah node akan mengklasifikasikan data *trainning*.

Kelebihan dari algortima decision tree adalah, yaitu kemampuannya untuk menghasilkan rule yang dapat diinspeksi dan di modifikasi oleh manusia. Sehingga metode ini cukup dapat diandalkan apabila terdapat kebutuhan untuk

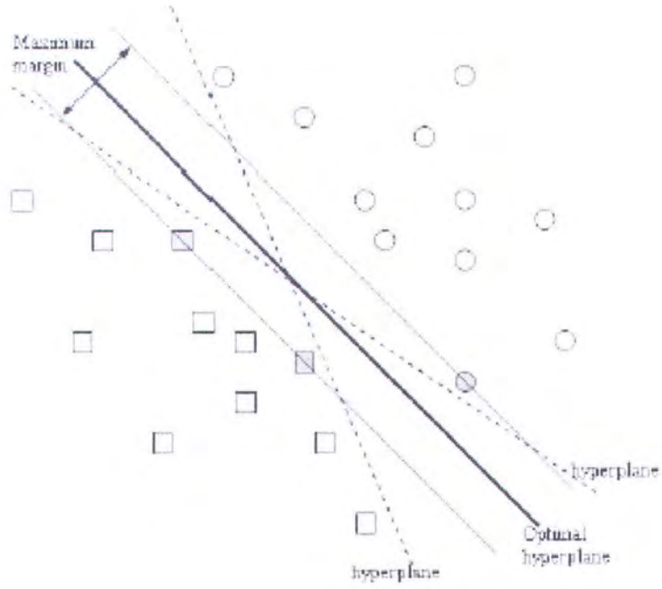
MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

menghasilkan serangkaian aturan / rule dasar, yang nantinya akan diperbaiki keakuratannya.

3.2.2 CLASSIFIER DALAM ORACLE TEXT 10.1

Dalam oracle text 10.1 ditambahkan algoritma baru (selain algoritma *decision tree* yang tetap ada dalam oracle 10.1) untuk menjalankan proses trainning dalam klasifikasi dokumen. Algortima tersebut adalah Support Vector Machine (SVM). SVM adalah algortima yang diturunkan dari teori statistik. Keuntungan dari penggunaan algoritma ini adalah kemampuannya untuk melakukan proses pembelajaran walau hanya dengan data trainning yang sedikit.

Support Vector Machine (SVM) adalah algoritma pembelajaran yang berdasar kepada sebuah fungsi inti (kernel) yang diperkenalkan pertama kali oleh Vapnik [Vap-1995]. *SVM classifier* pertama kali digunakan untuk menyelesaikan permasalahan pengenalan pola pada 2 kelas menggunakan prinsip *Structural Risk Minimization* [Bur-1998]. Data trainning direpresentasikan dalam sebuah ruang vektor, dan SVM akan mencari hyperplane terbaik yang akan memisahkan dua kelas yang ada. Kualitas dari hyperplane penentu ini ditentukan berdasarkan jarak (*hard* atau *soft margin*) diantara dua hyperplane yang dibentuk oleh vektor pendukung (*support vector*). *Hyperplane* penentu yang terbaik adalah *hyperplane* yang mampu memperbesar *margin* . Gambar 3.1 menjelaskan konsep ini dalam ruang berdimensi dua.



Gambar 3.1 Konsep dasar SVM dalam ruang berdimensi dua

3.3 LANGKAH – LANGKAH MENGGUNAKAN ORACLE TEXT

Sebelum menggunakan Oracle Text, ada beberapa hal yang sudah harus dipersiapkan terlebih dahulu, seperti menginstall basisdata Oracle 10g yang secara otomatis juga menginstall Oracle Text, karena Oracle Text merupakan satu *core* dengan basisdata Oracle, lalu meng-*unlock* skema CTX_SYS dalam basisdata Oracle tersebut. Ada beberapa tahapan dalam memulai suatu aplikasi pengklasifikasian dokumen pada Oracle Text yaitu :

1. Mempopulasi tabel dokumen
2. Membuat *index* pada tabel dokumen
3. Melakukan proses *training* dengan CTX_CLS.TRAIN pada koleksi dokumen yang sudah terklasifikasikan

4. Melakukan proses *clustering* dengan package CTX_CLUSTER jika koleksi dokumen belum terklasifikasikan.

Tahap-tahap tersebut akan dijelaskan pada subbab-subbab berikut ini.

3.4 MEMPOPULASI TABEL DOKUMEN

Sebelum bisa melalukan proses pengindeksan, dokumen tabel teks harus dipopulasi/ diisi terlebih dulu. Ada beberapa cara untuk meload teks ataupun rich dokumen ke kolom teks [Orb-2003], yaitu dengan:

- SQL INSERT
- SQL*Loader
- Prosedur PL/SQL insert dengan EMPTY_CLOB atau EMPTY_BLOB

3.4.1 SQL INSERT

Cara sederhana untuk mempopulasi tabel teks adalah dengan membuat suatu tabel dengan 2 kolom, `id` dan `text`, menggunakan `CREATE TABLE` dan kemudian menggunakan pernyataan `INSERT` untuk meload data. Sayangnya cara ini hanya bisa digunakan untuk mengisi kolom tabel dengan teks secara manual. Karena itu, untuk meload data file ke kolom tabel bertipe data CLOB dan BLOB akan dijelaskan pada subbab berikutnya. Contoh dibawah ini membuat kolom `id` sebagai primary key, yang opsional dan kolom `text` berupa `VARCHAR2`:

```
create table docs (id number primary key, text varchar2(80));
```

Untuk mempopulasi kolom `text`, digunakan pernyataan `INSERT` sebagai berikut:

```
insert into docs values(1,'it's the text of the first document');
insert into docs values(12,'it's the text of the second document');
```

3.4.2 SQL*LOADER

SQL*Loader dapat meload seluruh file kedalam internal LOBs. Contoh berikut menunjukkan bagaimana menggunakan SQL*Loader untuk meload berbagai format dokumen dari sistem operasi ke kolom BLOB.

Contoh berikut mempunyai 2 langkah:

- Membuat tabel

Contoh ini meload ke tabel articles_formatted yang dibuat sebagai berikut :

```
1  CREATE TABLE articles_formatted (
2    ARTICLE_ID    NUMBER PRIMARY KEY ,
3    AUTHOR        VARCHAR2(30),
4    FORMAT        VARCHAR2(30),
5    PUB_DATE      DATE,
6    TITLE         VARCHAR2(256),
7    TEXT          BLOB
8  );
```

Kolom article_id merupakan primary key. Dokumen diload dalam kolom text, yang bertipe BLOB.

- Membuat perintah SQL*Loader pembaca control file

Perintah berikut memulai loader, yang membaca control file LOADER1.DAT:

```
sqlldr userid=demo/demo control=loader1.dat log=loader.log
```

SQL*Loader control file menentukan kolom tempat yang akan diload dan mengintruksikan loader untuk meload data perbaris dari loader2.dat ke tabel

articles_formatted. Tiap baris di loader2.dat menyimpan list field-field yang dipisahkan oleh suatu koma untuk diload.

Contoh Control File : loader1.dat

```
1 load data
2 INFILE 'loader2.dat'
3 INTO TABLE articles_formatted
4 APPEND
5 FIELDS TERMINATED BY ','
6 (article_id SEQUENCE (MAX,1),
7  author CHAR(30), format, pub_date SYSDATE,
8  title, ext_fname FILLER CHAR(80),
9  text LOBFILE(ext_fname) TERMINATED BY EOF)
```

File ini menginstuksikan loader untuk meload data dari loader2.dat ke tabel articles_formatted dengan cara berikut :

1. posisi nomer urut pada baris menunjukkan field-field dokumen di loader2.dat yang ditulis ke kolom article_id.
2. field pertama di baris ditulis ke kolom author.
3. field kedua di baris ditulis ke kolom format.
4. tanggal saat itu yang diberikan oleh SYSDATE ditulis ke kolom pub_date.
5. judul dokumen, yang merupakan field ketiga dibaris, ditulis ke kolom title.
6. nama tiap dokumen yang diload dibaca ke variabel sementara ext_fname, dan dokumen aktual diload ke kolom text BLOB:

Contoh Data File :loader2.dat

```
1 Ben Kanobi, plaintext,Kawasaki news
2 rticle,../sample_docs/kawasaki.txt,
```

```
3   Joe Bloggs, plaintext,Java plug-
    in,../sample_docs/javaplugin.txt,
4
    John Hancock, plaintext,Declaration of
5
    Independence,../sample_docs/indep.txt,
6
    M. S. Developer, Word7,Newsletter
7
     example,../sample_docs/newsletter.doc,
8
    M. S. Developer, Word7,Resume
9   example,../sample_docs/resume.doc,
10  X. L. Developer, Excel7,Common
    example,../sample_docs/common.xls,
11
    X. L. Developer, Excel7,Complex
12  example,../sample_docs/solvsamp.xls,
13  Pow R. Point, Powerpoint7,Generic
14  presentation,../sample_docs/generic.ppt,
15  Pow R. Point, Powerpoint7,Meeting
16  presentation,../sample_docs/meeting.ppt,
17  Java Man, PDF,Java Beans paper,../sample_docs/j_bean.pdf,
18  Java Man, PDF,Java on the server
19  paper,../sample_docs/j_svr.pdf,
20  Ora Webmaster, HTML,Oracle home
    page,../sample_docs/oramnu97.html,
21  Ora Webmaster, HTML,Oracle Company
22  Overview,../sample_docs/oraoverview.html,
23  John Constable, GIF,Laurence J. Ellison :
24  portrait,../sample_docs/larry.gif,
25  Alan Greenspan, GIF,Oracle revenues :
    Graph,../sample_docs/oragraph97.gif,
    Giorgio Armani, GIF,Oracle Revenues :
    Trend,../sample_docs/oratrend.gif,
```

File ini berisi data yang akan diload ketiap baris tabel `articles_formatted`. Tiap baris berisi list field-field yang dipisahkan koma

yang akan diload ke articles_formatted. Field terakhir dari tiap nama baris file akan diload ke kolom teks.

3.4.3 PROSEDUR PL/SQL INSERT DENGAN EMPTY_CLOB ATAU EMPTY_BLOB

Sebelum menulis data ke internal LOB, buat direktori pointer yang akan mempointer Oracle untuk menunjuk langsung ke directory data yang akan diambil tersebut. Dengan perintah sebagai berikut :

```
create or replace directory dir_name as 'dir_location';
```

Lalu buat kolom LOB non-null, yang artinya kolom LOB harus berisi locator yang mempoint ke nilai empty atau populated LOB. Nilai kolom BLOB bisa diinisialisasi dengan menggunakan fungsi EMPTY_BLOB() sebagai predikat default. Sama saja, nilai suatu kolom CLOB atau NCLOB dapat diinisialisasikan dengan menggunakan fungsi EMPTY_CLOB().

Berikut ini merupakan contoh langkah-langkah prosedur insert data file ALGORITHMS.DOC yang ada di direktori E:\NUGIE\TA\DATA ke kolom dok di tabel docs

1. Buat direktori yang menginisialisasi direktori tersebut sebagai lokasi data

```
create or replace directory MY_DIR as 'E:\NUGIE\TA\DATA';
```

2. Menjalankan sintaks yang akan memasukkan file ALGORITHMS.DOC ke dalam kolom dok yang ada di tabel docs

```
1.   DECLARE
2.       f_lob   BFILE;
```

```
3.        b_lob   BLOB;
4.    BEGIN
5.     INSERT INTO DOCS(IDX,NAMADOC,DOK) VALUES
6.     ( 1,'ALGORITHMS.DOC',EMPTY_BLOB() )
7.     RETURN DOK INTO b_lob;
8.    f_lob := BFILENAME( 'MY_DIR', 'ALGORITHMS.DOC' );
9.    dbms_lob.fileopen(f_lob, dbms_lob.file_readonly);
10.   dbms_lob.loadfromfile(b_lob,f_lob,dbms_lob.getlength(f_lob));
11.   dbms_lob.fileclose(f_lob);
12.   COMMIT;
```

3.5 MEMBUAT INDEKS PADA TABEL DOKUMEN

Untuk membangun aplikasi apapun dengan menggunakan Oracle Text, yang harus dilakukan pertama kali adalah membuat indeks dari data yang telah diload kedalam tabel dokumen.

3.5.1 TIPE INDEKS PADA ORACLE TEXT

Ada beragam tipe indeks pada Oracle Text, yaitu : CONTEXT, CTXCAT dan CTXRULE. Tiap indeks tersebut memiliki spesifikasi sendiri-sendiri. Tabel dibawah ini menjelaskan tentang masing-masing tipe indeks

Tabel 3.1 Index Type Overview

Tipe Index	Tipe Aplikasi	Operator Query
CONTEXT	Digunakan untuk membangun aplikasi pencarian teks ketika teksnya terdiri dari dokumen-dokumen besar yang berkaitan.	CONTAINS

Tipe Index	Tipe Aplikasi	Operator Query
	Beberapa format dokumen yang bisa diindex dengan tipe index ini adalah Microsoft Word, HTML, XML, atau dokumen teks sederhana.	
CTXCAT	Di gunakan untuk mengindeks bagian – bagian text yang kecil, seperti nama, harga dan deskripsi yang disimpan di dalam kolom - kolom.	CATSEARCH
CTXRULE	Digunakan untuk membangun aplikasi klasifikasi dokumen. Index CTXRULE adalah index yg dibuat di tabel query dimana masing – masing query memiliki klasifikasi. Satu dokumen (HTML, XML, TXT) dapat diklasifikasi menggunakan operator MATCHES.	MATCHES

Karena spesifikasi terlengkap dimiliki oleh tipe indeks CONTEXT, maka dalam tugas akhir ini digunakan tipe indeks CONTEXT.

3.5.1.1 TIPE INDEKS CONTEXT

Tipe indeks ini digunakan untuk membuat indeks pada kolom teks. Indeks ini bisa diquery dengan operator CONTAINS di klausa WHERE pada pernyataan SELECT. Indeks ini membutuhkan sinkronisasi manual setelah proses DML. Sintaks untuk membuat indeks CONTEXT adalah sebagai berikut :

```
CREATE INDEX [schema.]index on [schema.]table(column)
 INDEXTYPE IS
```

```
ctxsys.context [ONLINE]
LOCAL [(PARTITION [partition] [PARAMETERS('paramstring')]
[, PARTITION [partition] [PARAMETERS('paramstring')]])]
[PARAMETERS(paramstring)] [PARALLEL n] [UNUSABLE];
```

Sedangkan sintaks untuk melakukan sinkronisasi pada tabel adalah sebagai berikut :

```
ctx_ddl.sync_index(
 idx_name  IN  VARCHAR2 DEFAULT NULL
 memory IN VARCHAR2 DEFAULT NULL,
 part_name IN VARCHAR2 DEFAULT NULL
 parallel_degree IN NUMBER DEFAULT 1 );
```

Tabel bisa secara opsional berisi primary key jika ingin mengidentifikasikan *rows* (baris) saat menggunakan prosedur di CTX_DOC. Jika tabel tidak memiliki primary key, maka *document services* akan mengidentifikasikan dokumen dengan ROWID.

Kolom yang dispesifikasikan harus salah satu dari tipe data berikut : CHAR, VARCHAR, VARCHAR2, BLOB, CLOB, BFILE, XMLType, or URIType.

DATE, NUMBER, dan kolom tabel bersarang tidak bisa diindeks. Kolom obyek juga tidak bisa diindeks, tapi atributnya bisa diindeks karena tipedatanya yang atomik.

Pengindeksan pada kolom ganda tidak didukung dengan tipe indeks CONTEXT. Hanya satu kolom yang bisa dispesifikasikan pada list kolom.

Saat membuat sebuah indeks dan tidak menspesifikasikan klausa parameter apapun, sebuah indeks akan dibuat dengan parameter default. Sebagai contoh,

command berikut membuat sebuah indeks CONTEXT dengan parameter default bernama myindex pada kolom text di tabel docs:

```
CREATE INDEX myindex ON docs(text) INDEXTYPE IS CTXSYS.CONTEXT;
```

Jika membuat index bertipe CONTEXT, operator CONTAINS harus digunakan untuk membuat query. Dengan operator CONTAINS, sejumlah operator dapat digunakan untuk mendefinisikan kriteria pencarian. Operator tersebut memungkinkan untuk membuat pencarian logika, proksimitas, fuzzy, stemming, thesaurus dan wildcard. Dengan indeks yang terkonfigurasi dengan benar, juga dapat dibuat pencarian section dalam dokumen yang memiliki struktur internal seperti HTML dan XML. Karena beragamnya dokumen dan begitu kompleksnya proses yang akan dilakukan pada dokumen maka pada aplikasi tugas akhir ini akan menggunakan operator query CONTAINS untuk proses testing dokumennya.

Pada pernyataan SELECT, spesifikasikan query pada klausa where dengan operator CONTAINS. Spesifikasikan pula operator SCORE untuk mengembalikan nilai dari tiap hit pada hitlist. Contoh berikut menunjukkan bagaimana membuat suatu query:

```
SELECT SCORE(1) title from news
WHERE CONTAINS(text,'oracle',1)> 0;
```

Hasil dari dokumen berskor tertinggi sampai ke dokumen berskor terendah dapat diurutkan dengan menggunakan klausa ORDER BY sebagai berikut:

```
SELECT SCORE(1), title from news
WHERE CONTAINS(text, 'oracle', 1) > 0 ORDER BY SCORE(1) DESC;
```

Dalam aplikasi PL/SQL, suatu cursor dapat digunakan untuk mengambil hasil dari query. Contoh berikut menggunakan query CONTAINS pada tabel NEWS

untuk mencari semua artikel yang berisi kata *oracle*. Judul dan skor dari 10 hit dokumen pertama merupakan outputnya.

```
1.  declare
2.  rowno number := 0;
3.  begin
4.  for c1 in (SELECT SCORE(1) score, title FROM news
5.  WHERE CONTAINS(text, 'oracle', 1) > 0
6.  ORDER BY SCORE(1) DESC)
7.  loop
8.  rowno := rowno + 1;
9.  dbms_output.put_line(c1.title||': '||c1.score);
10. exit when rowno = 10;
11. end loop;
12. end;
```

Contoh diatas menggunakan suatu cursor untuk melakukan perulangan untuk mendapatkan 10 hit pertama. Suatu alias *score* dideklarasikan untuk return value dari operator `SCORE`. Skor dan judul merupakan output ke standar output menggunakan notasi dot cursor.

Suatu query terstruktur, juga disebut sebagai query campuran adalah query yang memiliki predikat `CONTAINS` untuk meng-query suatu kolom teks dan memiliki predikat lain untuk meng-query kolom data terstruktur. Untuk membuat query terstruktur, klausa terstruktur dapat dispesifikasikan pada kondisi `WHERE` di pernyataan `SELECT`.

Sebagai contoh, pernyataan `SELECT` berikut mengembalikan semua artikel yang berisi kata *oracle* yang ditulis pada atau sesudah 1 Oktober 1997:

```
SELECT SCORE(1), title, issue_date from news
WHERE CONTAINS(text, 'oracle', 1) > 0
AND issue_date >= ('01-OCT-97')
ORDER BY SCORE(1) DESC;
```

3.5.1.2 TIPE INDEKS CTXCAT

Indeks CTXCAT adalah sebuah indeks yang dioptimasi untuk query campuran. Tipe indeks ini dapat dibuat saat menyimpan dokumen-dokumen kecil atau penggalan teks yang berhubungan dengan informasi struktur. Untuk meng-query indeks ini, gunakan operator CATSEARCH dan specifikasikan klausa terstruktur, jika ada. Performa query dengan indeks CTXCAT biasanya bekerja lebih baik untuk query terstruktur daripada dengan indeks CONTEXT.

3.5.1.3 TIPE INDEKS CTXRULE

Indeks CTXRULE dibuat untuk membangun aplikasi klasifikasi dokumen dimana dokumen yang datang akan langsung diklasifikasikan berdasarkan isinya. Aturan pengklasifikasi dapat didefinisikan sebagai query yang nantinya akan diindeks. Operator MATCHES dapat digunakan untuk mengklasifikasikan dokumen tunggal.

Indeks ini berisi suatu kolom berisi set query. Indeks ini dapat diquery dengan operator MATCHES di klausa WHERE pada pernyataan SELECT. Sintaks untuk membuat indeks CTXRULE adalah sebagai berikut :

```
CREATE INDEX [schema.]index on [schema.]table(column)
INDEXTYPE IS ctxsys.ctxrule
[PARAMETERS ('[lexer lexer_pref] [storage storage_pref]
```

```
[section group section_pref] [wordlist wordlist_pref]');
[PARALLEL n];
```

Kolom yang dispesifikasikan saat membuat indeks CTXRULE harus berupa VARCHAR2 atau CHAR2. Tipe lain tidak didukung oleh CTXRULE. Gunakan CREATE INDEX untuk membuat index CTXRULE. Parameter lexer, storage, section group, dan wordlist bisa dispesifikasikan jika diperlukan dan jika tidak, sistem akan menentukan spesifikasinya secara default. Tapi parameter filter, memory, datastore, stoplist, dan [no]populate parameter tidak bisa diaplikasikan ke tipe CTXRULE. Contoh membuat tipe index CTXRULE tanpa parameter adalah sebagai berikut:

```
CREATE INDEX ON myqueries(query) INDEXTYPE IS CTXSYS.CTXRULE;
```

Dengan adanya index CTXRULE yang dibuat pada query set, operator MATCHES dapat digunakan untuk mengklasifisikan dokumen. Asumsikan bahwa dokumen disimpan dalam tabel news:

```
CREATE TABLE news ( newsid NUMBER,author VARCHAR2(30),
source VARCHAR2(30), article CLOB );
```

Maka dapat dibuat trigger before insert trigger dengan operator MATCHES untuk meroute tiap dokumen ke tabel lain, news_route berdasarkan pada klasifikasinya:

```
1.  BEGIN
2.    -- find matching queries
3.    FOR c1 IN (select category
4.                 from myqueries
5.                where MATCHES(query, :new.article)>0)
6.    LOOP
```

```
7.      INSERT INTO news_route(newsid, category)
8.        VALUES (:new.newsid, c1.category);
9.    END LOOP;
10. END;
```

3.5.2 LANGKAH-LANGKAH MEMBUAT INDEKS

Langkah-langkah untuk membuat indeks adalah :

1. Menentukan dan membuat *preference* indeks, untuk menentukan preference *preprocessing* apa saja yang akan diberlakukan pada koleksi dokumen, jika tidak akan dikenakan default *indexing* oleh Oracle Text. *Preference* digunakan untuk menspesifikasikan informasi indeks, seperti dimana file-file yang akan diindeks akan disimpan dan metode yang digunakan untuk memfilter dokumen. Pembuatan preference menggunakan PL/SQL `CTX_DDL` package, yaitu dengan perintah `CTX_DDL.CREATE_PREFERENCE`. Setelah membuat preference, bisa diset atribut-atribut Tabel 3.2 menunjukkan macam-macam *preference* yang bisa diset dalam Oracle Text.

Tabel 3.2 Daftar Preference dalam Oracle Text

Preference	Deskripsi
Datastore	Menentukan jenis tempat penyimpanan dokumen
FILTER	Menentukan cara dokumen diubah ke teks sederhana
LEXER	Menentukan bahasa yang digunakan untuk mengindeks
WORDLIST	Menentukan bagaimana mendapatkan kata dasar dan

Preference	Deskripsi
Datastore	Menentukan jenis tempat penyimpanan dokumen
FILTER	Menentukan cara dokumen diubah ke teks sederhana
	menentukan proses query nantinya
STORAGE	Menentukan cara menyimpan hasil index
STOPLIST	Menentukan kata-kata atau tema yang tidak ikut diindeks
SECTION GROUP	Menentukan bagaimana mendefinisikan section dari dokumen

Pada subbab berikutnya akan dijelaskan lebih mendetil tentang macam-macam *preference* yang disediakan oleh Oracle Text.

2. Membuat indeks teks dengan menggunakan perintah SQL `CREATE INDEX`, memberi nama indeks, dan menspesifikasikan preference

3.5.3 JENIS – JENIS INDEX PREFERENCE

Berikut ini akan dijelaskan secara lebih mendetail, jenis – jenis *index preference*

3.5.3.1 DATASTORE PREFERENCE

Datastore Preference menspesifikasikan cara penyimpanan dokumen pada Oracle Text. Tabel 3.3 dibawah ini menjelaskan tentang *preference datastore* yang disediakan oleh Oracle Text.

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

Tabel 3.3 Daftar Datastore Preference

Tipe Datastore	Digunakan Jika
DIRECT_DATASTORE	Data disimpan dalam kolom teks. Tiap baris diindek sebagai dokumen tunggal.
MULTI_COLUMN_DATA STORE	Data disimpan dalam tabel teks dilebih dari 1 kolom. Kolom dirangkai membentuk suatu dokumen virtual
DETAIL_DATASTORE	Data disimpan dalam kolom teks. Dokumen terdiri dari 1 atau lebih baris disimpan dalam kolom teks ditabel detail, dengan informasi header disimpan di tabel master.
FILE_DATASTORE	Data disimpan diluar sistem operasi file. Nama file disimpan dalam kolom teks, 1 per baris.
NESTED_DATASTORE	Data disimpan dalam tabel bersarang.
URL_DATASTORE	Data disimpan diluar dalam file yang bertempat diintranet atau Internet. Uniform Resource Locators (URLs) disimpan dalam kolom teks.
USER_DATASTORE	Dokumen disintesis waktu diindeks oleh prosedur yan

Tipe Datastore	Digunakan Jika
	didefinisikan sendiri oleh pengguna.

Contoh membuat preference `MULTI_COLUMN_DATASTORE` adalah sebagai berikut :

```
begin
ctx_ddl.create_preference('my_multi','MULTI_COLUMN_DATASTORE');
ctx_ddl.set_attribute('my_multi','columns','column1,column2,column
3');
end;
```

3.5.3.2 FILTER PREFERENCE

Filter preference menspesifikasikan preference bagaimana teks akan difilter untuk proses *indexing*. Filter membuat dokumen terformat seperti DOC, PDF, PS ataupun dokumen plain text, HTML, dan XML agar bisa diindeks.

Untuk dokumen terformat, Oracle menyimpan dokumen dalam format yang dipahami dan menggunakan filter untuk membuat versi plain text atau HTML sementara untuk dokumen tersebut. Oracle mengindeks kata yang diambil dari versi plain text atau HTML tersebut untuk dokumen terformat.

Tabel dibawah ini menjelaskan tentang filter *preference datastore* yang disediakan oleh Oracle Text.

Tabel 3.4 List Preference Filter

Tipe Filter Preference	Deskripsi
CHARSET_FILTER	Filter pengubah Character set

Tipe Filter Preference	Deskripsi
INSO_FILTER	Inso filter for untuk memfilter dokumen terformat
NULL_FILTER	Tidak butuh pemfilteran. Digunakan untuk mengindeks dokumen TEXT, HTML, atau XML
USER_FILTER	Filter eksternal yang didefinisikan pengguna untuk pemfilteran khusus
PROCEDURE_FILTER	Prosedur filter yang dapat didefinisikan sendiri oleh pengguna untuk digunakan pada pemfilteran khusus.

Contohnya, untuk melakukan indexing pada sekumpulan dokumen HTML bisa dengan menspesifikasikan preference dengan `NULL_FILTER` dan `HTML_SECTION_GROUP`, seperti berikut :

```
create index myindex on docs(htmlfile)
indextype is ctxsys.context parameters('filter ctxsys.null_filter
section group ctxsys.html_section_group');
```

3.5.3.3 Lexer Preference

Lexer preference digunakan untuk menentukan analisa leksikal untuk bahasa tertentu pada teks yang akan diindeks. Tabel dibawah ini menjelaskan tentang *lexer preference* yang disediakan oleh Oracle Text.

Tabel 3.5 List Preference Lexer

Tipe	Deskripsi
BASIC_LEXER	Mengekstrak token bahasa Inggris dan bahasa

Tipe	Deskripsi
	Eropa Barat pada umumnya yang memakai spasi sebagai pembatas kata.
MULTI_LEXER	Mengekstrak tabel dokumen bermulti bahasa
CHINESE_VGRAM_LEXER	Mengekstrak token teks bahasa Cina.
CHINESE_LEXER	Mengekstrak token teks bahasa Cina
JAPANESE_VGRAM_LEXER	Mengekstrak token teks bahasa Jepang
JAPANESE_LEXER	Mengekstrak token teks bahasa Jepang
KOREAN_LEXER	Mengekstrak token teks bahasa Korea
KOREAN_MORPH_LEXER	Mengekstrak token teks bahasa Korea (direkomendasikan)
USER_LEXER	Lexer yang pengguna buat untuk mengekstrak bahasa tertentu.

Contoh berikut mengeset karakter *printjoin* dan men-disable pengindeksan tema dengan `BASIC_LEXER`:

```
1 begin
2 ctx_ddl.create_preference('mylex', 'BASIC_LEXER');
3 ctx_ddl.set_attribute('mylex', 'printjoins', '_-');
4 ctx_ddl.set_attribute ( 'mylex', 'index_themes','NO');
5 ctx_ddl.set_attribute ( 'mylex', 'index_text', 'YES');
6 end;
```

3.5.3.4 WORDLIST PREFERENCE

Wordlist preference digunakan untuk memungkinkan opsi query seperti stemming, dan fuzzy matching untuk bahasa Inggris, Belanda, Perancis, Jerman, Italia dan Spanyol. *Wordlist preference* juga dapat digunakan untuk memungkinkan pengindeksan substring dan *prefix*(awalan) yang akan memperbaiki performa pada query wildcard dengan `CONTAINS` dan `CATSEARCH`.

Untuk membuat wordlist preference, digunakan `BASIC_WORDLIST`, yang hanya merupakan satu tipe yang tersedia. `BASIC_WORDLIST` mempunyai atribut-atribut sebagaimana yang tercantum pada tabel 3.5 berikut ini.

Tabel 3.6 Atribut `BASIC_WORDLIST`

Atribut	Nilai Atribut
stemmer	Menspesifikasikan stemmer bahasa yang digunakan, yaitu: NULL (tanpa stemming) ENGLISH (Bahasa Inggris Berinfleksi) DERIVATIONAL (Bahasa Inggris Asli) DUTCH FRENCH GERMAN ITALIAN SPANISH AUTO (otomatis mendeteksi bahasa untuk stemming,

Atribut	Nilai Atribut
	sesuai setting bahasa saat instalasi)
fuzzy_match	Menspesifikasikan fuzzy matching cluster yang akan digunakan, yaitu: GENERIC JAPANESE_VGRAM KOREAN CHINESE_VGRAM ENGLISH DUTCH FRENCH GERMAN ITALIAN SPANISH OCR AUTO (otomatis mendeteksi bahasa untuk stemming, sesuai setting bahasa saat instalasi)
fuzzy_score	Menspesifikasikan default batas terbawah skor fuzzy, dengan nilai antara 0 sampai 80. Teks bernilai dibawah itu

Atribut	Nilai Atribut
	takkan ditampilkan. Nilai default 60. Semakin rendah spesifikasi nilai fuzzy_score, semakin banyak kata cocok yang dihasilkan, begitu pula sebaliknya.
fuzzy_numresults	Menspesifikasikan nilai maksimum untuk ekspansi fuzzy, yaitu antara 1 sampai 5000. Nilai default 100. Semakin rendah nilai fuzzy_numresult yang dispesifikasikan semakin banyak kata-kata cocok yang dihasilkan. Begitu pula sebaliknya.
substring_index	Menspesifikasikan TRUE untuk membuat substring index. Substring index memperbaiki query wildcard *left-truncated* dan *double-truncated* seperti *%ing* or *%benz%*. Nilai default FALSE.
prefix_index	Menspesifikasikan TRUE untuk memungkinkan pengindeksan prefiks. Pengindeksan prefiks memperbaiki performa untuk pencarian wildcard *right truncated* seperti TO%. Nilai defaults FALSE.
prefix_length_min	Menspesifikasikan panjang minimum dari indeks prefiks. Nilai default 1.
prefix_length_max	Menspesifikasikan panjang maksimum dari indeks prefiks. Nilai default 64.

Atribut	Nilai Atribut
wildcard_maxterms	Menspesifikasikan jumlah kata maksimum pada ekspansi wildcard. Gunakan nilai antara 1 sampai 15,000. nilai default 5,000.

Contoh berikut memungkinkan stemming dan fuzzy matching untuk bahasa Inggris. Preference `STEM_FUZZY_PREF` mengeset jumlah ekspansi ke nilai maksimum yang diijinkan. Preference ini juga mengintruksikan pada sistem untuk membuat indeks substring untuk memperbaiki performa pencarian *double-truncated.*

```
1  begin
2  ctx_ddl.create_preference('STEM_FUZZY_PREF','BASIC_WORDLIST');
3  ctx_ddl.set_attribute('STEM_FUZZY_PREF','FUZZY_MATCH','ENGLISH');
4  ctx_ddl.set_attribute('STEM_FUZZY_PREF','FUZZY_SCORE','0');
5  ctx_ddl.set_attribute('STEM_FUZZY_PREF','FUZZY_NUMRESULTS','5000');
6  ctx_ddl.set_attribute('STEM_FUZZY_PREF','SUBSTRING_INDEX','TRUE');
7  ctx_ddl.set_attribute('STEM_FUZZY_PREF','STEMMER','ENGLISH');
9  end;
```

3.4.2.6 Storage Preference

Storage preference digunakan untuk menspesifikasikan *tablespace* dan pembuatan parameter untuk tabel-tabel yang berhubungan dengan teks indeks. Sistem menyediakan tipe penyimpanan tunggal yang disebut `BASIC_STORAGE`. `BASIC_STORAGE` punya atribut-atribut sebagaimana yang tercantum pada tabel 3.6 dibawah ini:

Tabel 3.7 Atribut BASIC_STORAGE

Atribut	Nilai Atribut
i_table_clause	Klausa parameter pembuatan tabel dr$*indexname*$I. Menspesifikasikan klausa penyimpanan dan tablespace untuk ditambahkan pada akhir pernyataan internal CREATE TABLE. Tabel I adalah data tabel indeks.
k_table_clause	Klausa parameter pembuatan tabel dr$*indexname*$K. Menspesifikasikan klausa penyimpanan dan tablespace untuk ditambahkan pada akhir pernyataan internal CREATE TABLE. Tabel K adalah tabel keymap.
r_table_clause	Klausa parameter pembuatan tabel dr$*indexname*$R. Menspesifikasikan klausa penyimpanan dan tablespace untuk ditambahkan pada akhir pernyataan internal CREATE TABLE. Tabel R adalah tabel rowid. Klausa defaultnya adalah: 'LOB(DATA) STORE AS (CACHE)'
n_table_clause	Klausa parameter pembuatan tabel dr$*indexname*$N. Menspesifikasikan klausa penyimpanan dan tablespace untuk ditambahkan pada akhir pernyataan internal CREATE TABLE. Tabel N adalah list tabel negatif.
i_index_clause	Klausa parameter pembuatan tabel dr$*indexname*$X. Menspesifikasikan klausa penyimpanan dan tablespace untuk ditambahkan pada akhir pernyataan internal CREATE TABLE.

Atribut	Nilai Atribut
	Klausa defaultnya adalah: 'COMPRESS 2' yang akan mengintruksikan pada Oracle untuk mengkompres tabel indeks ini. Jika memilih untuk mengesampingkan default, Oracle merekomendasikan mengikutkan COMPRESS 2 pada klausa parameter untuk mengkompres tabel ini, karena kopresi bisa menghemat ruang disk dan membantu performa query.
p_table_clause	Klausa parameter indeks substring jika meng-enable SUBSTRING_INDEX di BASIC_WORDLIST. Menspesifikasikan klausa penyimpanan dan tablespace untuk ditambahkan pada akhir pernyataan internal CREATE TABLE. Tabel P adalah tabel pengorganisasi indeks sehingga klausa penyimpanan yang dispesifikasikan harus sesuai dengan tipe tabel ini.

Contoh berikut menspesifikasikan bahwa tabel indeks dibuat di tablespace foo dengan luas diinisialkan 1K:

```
1  begin
2  ctx_ddl.create_preference('mystore', 'BASIC_STORAGE');
3  ctx_ddl.set_attribute('mystore','I_TABLE_CLAUSE','tablespace
4  foo
5  storage(initial 1K)');
```

```
6   ctx_ddl.set_attribute('mystore','K_TABLE_CLAUSE','tablespace
7   foo storage(initial 1K)');
8   ctx_ddl.set_attribute('mystore','R_TABLE_CLAUSE','tablespace
9   foo storage(initial 1K)');
10  ctx_ddl.set_attribute('mystore','N_TABLE_CLAUSE','tablespace
11  foo storage(initial 1K)');
12  ctx_ddl.set_attribute('mystore','I_INDEX_CLAUSE','tablespace
13  foo storage(initial 1K)');
14  ctx_ddl.set_attribute('mystore','P_TABLE_CLAUSE','tablespace
15  foo storage(initial 1K)');
16  end;
```

3.5.3.5 STOPLIST PREFERENCE

Stoplist adalah kata-kata pada bahasa tertentu yang tidak ikut diindeks. Dalam bahasa Inggris, *stopthemes* yang tidak ingin diindeks juga bisa ditentukan. Defaultnya, sistem mengindeks teks menggunakan stoplist yang disediakan sistem sehubungan dengan bahasa yang dispesifikasikan pada basisdata.

Oracle Text menyediakan *stoplist default* untuk banyak bahasa termasuk bahasa Inggris, Perancis, Jerman, Spanyol, Belanda, dan Denmark. *Stoplist* tersebut berisi list *stopword* yang akan dihilangkan dari proses *indexing.*

Stoplist multi-bahasa juga bisa dibuat untuk menangani *stopwords* untuk bahasa tertentu. *Stoplist* multi-bahasa akan berguna jika `MULTI_LEXER` digunakan untuk mengindeks dokumen dari berbagai bahasa, seperti bahasa Inggris, Jerman, dan Jepang.

Untuk membuat stoplist multi-bahasa, digunakan prosedur `CTX_DLL.CREATE_STOPLIST` dan harus menspesifikasikan tipe stoplist untuk

MULTI_STOPLIST. Stopword untuk bahasa tertentu juga bisa ditambah dengan CTX_DDL.ADD_STOPWORD.

Saat pengindeksan, kolom bahasa pada tiap dokumen diperiksa, dan hanya stopwords untuk bahasa tersebut yang dihilangkan. Di waktu query, setting session bahasa menentukan stopword aktif, seperti menentukan lexer yang aktif saat menggunakan multi-lexer.

Stoplist bisa dibuat sendiri dengan menggunakan prosedur CTX_DLL.CREATE_STOPLIST. Dengan prosedur ini suatu BASIC_STOPLIST stoplist bahasa tunggal atau MULTI_STOPLIST untuk stoplist multi-bahasa bisa dibuat.

Saat membuat stoplist sendiri, stoplist ini harus dispesifikasikan pada klausa parameter CREATE INDEX. Stoplist default selalu dinamakan CTXSYS.DEFAULT_STOPLIST.

Prosedur dibawah ini bisa digunakan untuk memodifikasi stoplist:

- CTX_DDL.ADD_STOPWORD
- CTX_DDL.REMOVE_STOPWORD
- CTX_DDL.ADD_STOPTHEME
- CTX_DDL.ADD_STOPCLASS

Saat memodifikasi CTXSYS.DEFAULT_STOPLIST dengan package CTX_DDL, indeks harus dibuat kembali supaya perubahan dapat memperlihatkan hasil

Stopword dapat ditambah secara dinamis pada stoplist default atau buatan sendiri dengan perintah ALTER INDEX. Saat menambah stopword secara dinamis, tidak perlu mengindeks ulang, karena kata tersebut akan langsung menjadi stopword dan dihilangkan dari indeks.

Walaupun stopword dapat ditambahkan secara dinamis pada indeks, stopword tidak dapat dihilangkan secara dinamis. Untuk menghilangkan stopword, harus digunakan `CTX_DDL.REMOVE_STOPWORD`, drop indeks dan create ulang. Contoh untuk membuat stoplist dengan nama mystop dan menambahkan stopword *of, no, where* :

```
1  begin
2  ctx_ddl.create_stoplist('mystop', 'BASIC_STOPLIST');
3  ctx_ddl.add_stopword('mystop', 'of');
4  ctx_ddl.add_stopword('mystop', 'no');
5  ctx_ddl.add_stopword('mystop', 'where');
6  end;
```

3.5.3.6 SECTION GROUP PREFERENCE

Untuk menggunakan query `WITHIN` pada bagian dokumen yang berstruktur seperti HTML dan XML, section group harus dibuat sebelum mendefinisikan bagian-bagian yang diinginkan. Spesifikasikan section group dalam klausa parameter `CREATE INDEX`. Untuk membuat section group, harus dispesifikasikan salah satu dari tipe group yang dijabarkan pada tabel 3.7 berikut ini dengan prosedur `CTX_DDL.CREATE_SECTION_GROUP`:

Tabel 3.8 Tipe-tipe grup pada `SECTION_GROUP`

Section Group Preference	Deskripsi
`NULL_SECTION_GROUP`	Digunakan saat mendefinisikan tanpa bagian atau saat hanya mendefinisikan bagian kalimat atau paragraf.

Section Group Preference	Deskripsi
	Merupakan default.
BASIC_SECTION_GROUP	Digunakan saat mendefinisikan bagian tag awal dan akhir dalam bentuk <A> and </A>. Tipe grup ini tidak mendukung input seperti tanda kurung yang tidak cocok, tag komentar, dan atribut. Gunakan HTML_SECTION_GROUP untuk tipe input ini.
HTML_SECTION_GROUP	Digunakan saat pengindeksan dokumen HTML dan pendefinisian bagian pada dokumen HTML.
XML_SECTION_GROUP	Digunakan saat pengindeksan dokumen XML dan pendefinisian bagian pada dokumen XML.
AUTO_SECTION_GROUP	Digunakan saat membuat zona bagian secara otomatis untuk tiap pasangan tag awal dan akhir pada dokumen XML. Nama bagian diambil tag XML dan juga *case sensitive* sebagaimana di XML. Atribut bagian juga dibuat otomatis untuk tag XML yang memiliki atribut. Bagian atribut dinamakan dalam bentuk attribute@tag. Stop section, tag yang kosong, intruksi pemrosesan dan komentar tidak diindeks. Pembatasan berikut diberlakukan untuk section group

Section Group Preference	Deskripsi
	otomatis: Tidak bisa menambah zona, field, atau sections khusus pada section group otomatis. Pembagian otomatis tidak mengindeks tipe dokumen XML (elemen root.) bagaimanapun, However, stop section dapat didefinisikan dengan tipe dokumen. Panjang tag yang diindeks, termasuk prefiks dan ruang nama, tidak boleh melebihi 64 karakter. Tag yang lebih panjang dari itu takkan diindeks.
PATH_SECTION_GROUP	Digunakan saat mengindeks dokumen XML. Mirip dengan AUTO_SECTION_GROUP. Perbedaannya adalah dengan section group ini, pencarian path dengan operator INPATH dan HASPATH bisa dilakukan. Query juga akan *case-sensitive* untuk tag dan nama atribut. Stop section tidak diijinkan.
NEWS_SECTION_GROUP	Digunakan saat mendefinisikan bagian dokumen terformat newsgroup sesuai dengan RFC 1036.

Contoh berikut membuat suatu section group yang disebut htmgroup dengan tipe grup HTML.

```
begin
ctx_ddl_create_section_group('htmgroup','HTML_SECTION_GROUP');
end;
```

3.6 MELAKUKAN PROSES *TRAINING* DENGAN CTX_CLS.TRAIN

Teori yang mendasari *training* (pembelajaran) pengklasifikasi adalah bahwa dokumen-dokumen yang berhubungan mempunyai kosa kata yang lazim dan khusus, yang ditiap kelompok dokumen, ada beberapa kata yang muncul pada hampir semua dokumen dalam kelompok tersebut dan yang muncul hanya secara sporadik dalam dokumen bukan dalam kelompok. Tugas pembelajaran adalah untuk menemukan kosa kata tersebut [Orc-2002].

Skema umum yang proses pembelajaran lakukan adalah :

1. Membaca tabel kategorisasi ke memori
2. *Scan* tabel $I untuk mendapatkan token-token yang paling umum dalam dokumen set
3. *Scan* list token-token dari langkah 2. untuk tiap token:
   a. Baca token_info untuk menghasilkan list doc_id berisi token
   b. Kombinasikan list dengan tabel kategorisasi dari langkah 1 untuk menghasilkan list kategori-kategori yang beridi dokumen berisi token
   c. Untuk tiap kategori di list, kalkulasikan skor kualitas yang mengekspresikan kekhususan token terhadap kelompoknya
   d. Akumulasikan token-token berskor tertinggi dalam kategori
4. Bangun bitmap dari semua token-token berskor tertinggi ditiap kategori dan di dokumen-dokumen mana token-token tersebut ditemukan.

5. Buat suatu decision tree untuk tiap kategori
6. Ekspresikan kembali decision tree sebagai seperangkat aturan-aturan

Untuk alasan performa, beberapa langkah yang terpisah diatas sebenarnya dilakukan pada waktu yang bersamaan, tapi secara konseptual benar. Pembelajaran bisa dilakukan dengan preference `SVM_CLASSIFIER`, dimana atribut-atributnya mengontrol berbagai langkah yang ditulis diatas. Semakin melihat proses lebih dekat, semua akan mencakup atribut-atribut tersebut. Membuat preference `SVM_CLASSIFIER` tidak berbeda dengan preference lain, misalnya:

```
1  ctx_ddl.create_preference('my_classifier','SVM_CLASSIFIER');
2  exec
3  ctx_ddl.set_attribute('my_classifier','MAX_DOCTERMS','50');
4  exec
5  ctx_ddl.set_attribute('my_classifier','MAX_FEATURES','3000');
6  exec ctx_ddl.set_attribute('my_classifier','THEME_ON','TRUE');
7  exec ctx_ddl.set_attribute('my_classifier','TOKEN_ON','TRUE');
8  exec ctx_ddl.set_attribute('my_classifier','STEM_ON','FALSE');
```

Jika preference tidak diset maka `CTX_CLS.TRAIN` akan dieksekusi dengan preference default. Sedangkan sintaks `CTX_CLS.TRAIN` adalah sebagai berikut:

```
1  CTX_CLS.TRAIN(
2   index_name in varchar2,
3   doc_id in varchar2,
4   cattab in varchar2,
5   catdocid in varchar2,
6   catid in varchar2,
7   restab in varchar2,
8   preference_name in varchar2 DEFAULT NULL);
```

Parameter – parameter masukan dalam prosedur tersebut, akan dijelaskan berikut ini :

1. index_name

digunakan untuk menentukan nama indeks dari dokumen, yang telah dibuat.

2. docid

untuk menentukan kolom doc ID dalam tabel dokumen

3. cattab

untuk menentukan nama tabel kategori (kelas)

4. catdocid

untuk menentukan kolom category ID dalam tabel category

5. restab

untuk menentukan nama tabel tempat menyimpan aturan – aturan yang berhasil dibuat oleh SVM_CLASSIFIER

6. preference_name

untuk menentukan nama prefence yang telah ditentukan sebelumnya.

Tabel Hasil (restab) yang digunakan untuk menyimpan aturan – aturan yang berhasil di-generate oleh SVM_CLASSIFIER, minimal harus terdiri atas tiga kolom cat_id dengan tipe number, type dengan tipe number, dan rule dengan tipe blob,

Nilai atribut tiap-tiap preference yang bisa diset pada SVM_CLASSIFIER dijabarkan pada tabel 3.9 berikut :

Tabel 3.9 Daftar Preference Pada SVM_Classifier

Nama Atribut	Tipe Data	Default	Nilai Min	Nilai Max	Deskripsi
MAX_DOCTEMRS	I	50	10	8192	Untuk menentukan jumlah *term* maksimal untuk merepresentasikan sebuah dokumen
MAX_FEATURES	I	3000	1	100.000	Untuk menentukan jumlah fitur maksimal
THEME_ON	B	FALSE	NULL	NULL	Untuk menentukan digunakannya atau tidaknya theme sebagai fitur indeks.
TOKEN_ON	B	TRUE	NULL	NULL	Untuk menentukan digunakan atau tidaknya token sebagai fitur indeks
STEM_ON	I	10	0	100	Untuk menentukan dilakukan atau tidaknya perubahan token menjadi bentuk dasar (*stemmed token*), ini hanya akan berlaku apabila index_trems diaktifkan pada

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

Nama Atribut	Tipe Data	Default	Nilai Min	Nilai Max	Deskripsi
					lexer

3.7 SUPERVISED CLASSIFICATION BERBASIS SVM

SVM merupakan salah satu jenis algoritma pembelajaran mesin yang diturunkan dari teori pembelajaran statistikal. Pebedaan *SVM classifier* dengan *classifier* yang lain di dalam oracle text (misalnya *decision tree classifier*) adalah :

- *Preference* yang digunakan pada saat pemanggilan prosedur `CTX.TRAIN` harus bertipe `SVM_CLASSIFIER` bukan `RULE_CLASSIFIER` . Jika perubahan atribut `CTX.TRAIN` tidak diinginkan, maka dapat juga digunakan preference `CTXSYS.SVM_CLASSIFIER`.
- *Context index* pada tabel dokumen tidak perlu dipopulasi, sehingga digunakan key word NOPOPULATE. *Classfier* menggunakan index ini hanya untuk menemukan sumber dari teks, yaitu datastore dan preference filter, juga untuk menentukan bagaimana memproses teks, melalui *lexer* dan *sectioner preference.*
- Tabel yang digunakan untuk menampung rule (hasil dari proses *trainning*), harus terdiri dari minimal 3 kolom berikut : cat_id number, type number dan rule bertipe blob.

Rule yang dihasilkan oleh *SVM Classifier* dituliskan dalam format *BLOB*, hal ini tentu saja akan menjadi sedikit hambatan bagi pengguna. Karena tidak seperti *rule* yang dihasilkan oleh *decision tree classifier*, karena rule dalam format

BLOB ini tidak dapat di rubah atau di modifikasi seandainya terdapat kesalahan dalam klasifikasi . Berikut ini adalah contoh klasifikasi berbasis SVM :

1. Membuat dan mempopulasi tabel dokumen untuk trainning

```
create table doc (id number primary key, text
varchar2(2000));
insert into doc values(1,'1 2 3 4 5 6');
insert into doc values(2,'3 4 7 8 9 0');
insert into doc values(3,'a b c d e f');
insert into doc values(4,'g h i j k l m n o p q r');
insert into doc values(5,'g h i j k s t u v w x y z');
```

2. membuat dan mempopulasi tabel *category*

```
create table testcategory (
doc_id number,cat_id number,cat_name varchar2(100));
insert into testcategory values (1,1,'number');
insert into testcategory values (2,1,'number');
insert into testcategory values (3,2,'letter');
insert into testcategory values (4,2,'letter');
insert into testcategory values (5,2,'letter');
```

3. Membuat indeks pada tabel dokumen, dengan tanpa harus dipopulasi.

```
create index docx on doc(text) indextype is ctxsys.context
parameters('nopopulate');
```

4. Set SVM_CLASSIFIER

```
Exec ctx_ddl.create_preference('my_classifier',
 'SVM_CLASSIFIER');
exec ctx_ddl.set_attribute('my_classifier',
```

```
'MAX_FEATURES','100');
```

5. Membuat table rule

```
create table restab (cat_id number, type number, rule
blob);
```

6. Jalankan proses trainning

```
exec ctx_cls.train('docx','id','testcategory',
'doc_id','cat_id','restab','my_classifier');
```

7. Buat index pada kolom rule di tabel rule

```
exec ctx_ddl.create_preference('my_filter', 'NULL_FILTER');
create index restabx on restab (rule) indextype is
ctxsys.ctxrule
parameters ('filter my_filter classifier my_classifier');
```

8. Melakukan testing terhadap rule yang dihasilkan

```
1  select cat_id, match_score(1) from restab
2  where matches(rule, '4 5 6',1)>50;
3  select cat_id, match_score(1) from restab
4  where matches(rule, 'f h j',1)>50;
5  drop table doc;
6  drop table testcategory;
7  drop table restab;
8  exec ctx_ddl.drop_preference('my_classifier');
9  exec ctx_ddl.drop_preference('my_filter');
```

BAB IV

PERANCANGAN DAN IMPLEMENTASI PERANGKAT LUNAK

BAB IV
PERANCANGAN DAN
IMPLEMENTASI PERANGKAT LUNAK

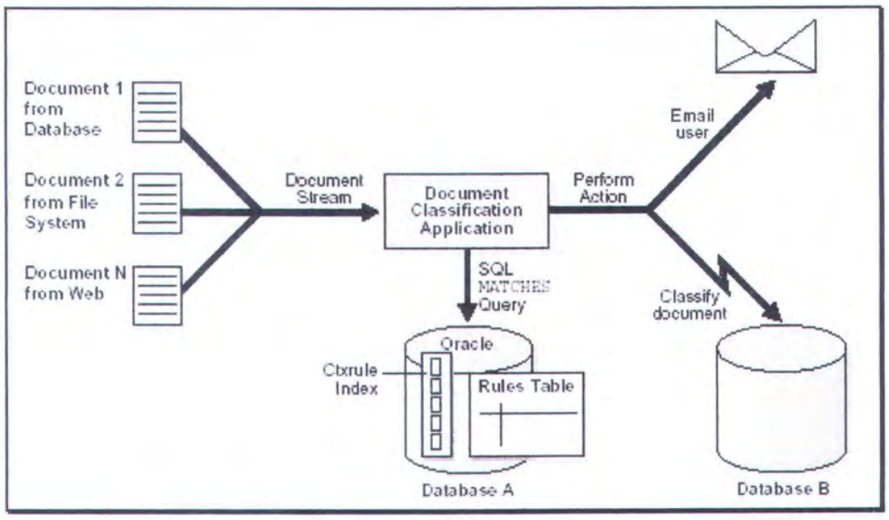
Pada bab ini akan dibahasa mengenai perancangan dan pembuatan perangkat lunak, SVM TEXT CLASSIFIER berbasis Oracle Text 10.1.0.2 ini. Bab ini dibagi menjadi dua bagian utama, yaitu bagian perancangan dan bagian implementasi. Bagian perancangan akan menjelaskan mengenai elemen – elemen penting dari aplikasi, bagaimana elemen – elemen tersebut direncanakan untuk dibuat. Sedangkan bagian implementasi akan menjelaskan bagaimana elemen – elemen tersebut akan diimplementasikan.

4.1 PERANCANGAN PERANGKAT LUNAK

Secara khusus aplikasi yang akan dibangun dalam tugas akhir ini disebut sebagai aplikasi pengklasifikasian dokumen, yang secara garis besar memiliki bagan proses sebagaimana yang terlihat pada gambar 4.1.

Dimana dalam aplikasi ini akan dimasukkan serangkaian dokumen sebagai data *input*. Dan aplikasi pengklasifikasi dokumen akan berusaha mengenali kelas dari dokumen masukan ini.

Untuk lebih menyederhanakan pembahasan, maka percangan perangkat lunak akan diarahkan kedalam 2 bagian utama dari aplikasi, yaitu, data, dan perancangan proses.



Gambar 4.1 Bagan proses aplikasi text classification

4.1.1 PERANCANGAN DATA

Tampak dalam skema diatas, bahwa terdapat tiga bagian yang berhubungan dengan penanganan data dalam aplikasi. Ketiga bagian itu antara lain data masukan, data proses dan data keluaran. Karena masing – masing jenis data tersebut harus disusun dalam struktur yang bersesuaian dengan proses yang akan dilakukan dalam tahap – tahap masukan, proses dan keluaran. Penyusunan ini sekaligus akan menjadi batasan dan atau fitur dari aplikasi yang akan dibuat.

4.1.1.1 DATA MASUKAN

Data masukan adalah data yang akan diolah oleh aplikasi ini dan dapat berupa data dokumen multiformat. File dokumen terserbut boleh berformat TXT, DOC (Microsoft Word) , PDF, PS, RTF dan sebagainya. Data ini harus diletakkan dalam drive yang ada dalam sistem komputer. Batasan yang diterapkan pada data masukan dalam aplikasi ini yaitu : dokumen – dokumen yang dimasukkan harus

sudah disertai dengan label kelasnya, mengingat aplikasi yang akan mengolah data tersebut nantinya menggunakan metode pembelajaran *supervised.*

4.1.1.2 DATA PROSES

Data proses adalah data – data yang dibutuhkan selama proses berjalan, data ini dapat berupa parameter – parameter atau atribut yang dibutukan oleh proses, untuk menentukan karakteristik dari proses yang akan dijalankan. Tentu saja setiap perubahan data proses akan sangat memperngaruhi data keluaran yang dihasilkan oleh aplikasi. Proses – proses dalam aplikasi ini yang membutuhkan parameter data yaitu :

1. proses Indexing, proses ini mememerlukan parameter data proses karena oracle text menyediakan berbagai macam tipe index yang dapat digunakan, dengan terlebih dahulu disesuaikan dengan kebutuhan dari si pemakai. Proses ini juga memerlukan data proses sebagai referensi, terutama dalam hal penggunaan stemming, penggunaan stopwords dan penggunaan thesaurus. Data stemming dan thesaurus sudah disediakan oleh oracle, sedangkan untuk stopword, dapat disediakannya dalam bentuk file teks yang terdiri atas daftar kumpulan kata yang dipisahkan dengan *carriage return.* Stopwors ini biasanya didapatkan dari hasil berbagai penelitian di internet.
2. proses validasi atau pendistribusian data, proses ini memerlukan parameter penentu banyaknya data yang akan digunakan dalam proses trainning dan testing.

3. proses training, proses ini memerlukan beberapa atribut data yaitu : MAX_DOCTERMS, dan MAX_FEATURES.

Dalam aplikasi ini ketiga jenis data proses diatas dijadikan satu dalam satu paket yang disebut sebagai 'Project', hal ini bertujuan agar nantinya pengguna dapat dengan mudah merubah – rubah data proses dengan maksud untuk melakukan analisa kemampuan dari aplikasi. Setiap aplikasi berjalan, pengguna dapat merilis beberapa *project* sekaligus yang berarti pengguna akan melakukan ujicoba klasifikasi dengan beberapa macam aturan setting data proses.

4.1.1.3 DATA KELUARAN

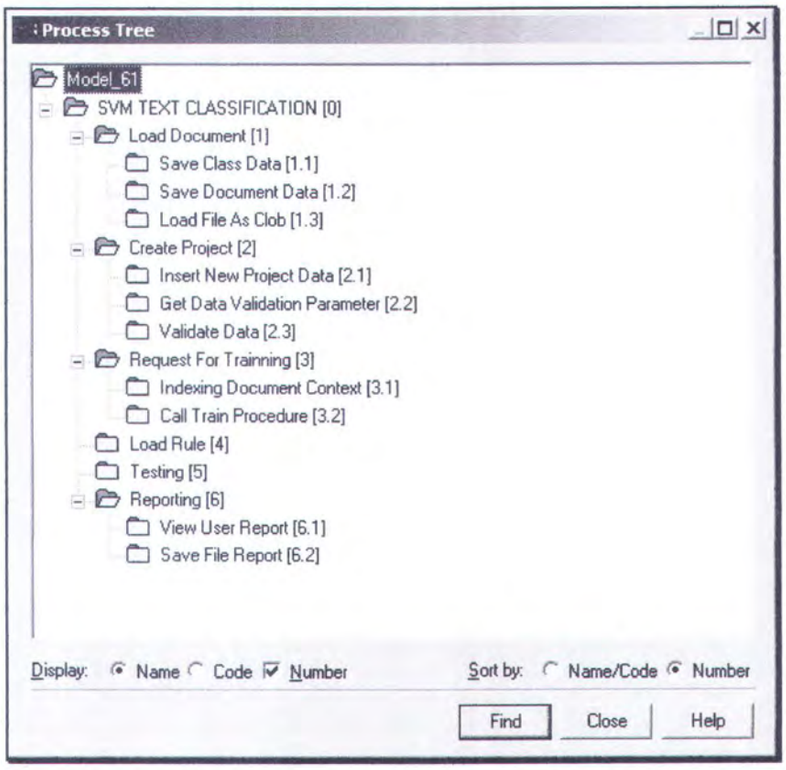
Data keluaran adalah data - data yang dihasilkan oleh aplikasi, yaitu antara lain :

1. Tampilan data dokumen yang digunakan dalam aplikasi
2. Tampilan penyebaran data untuk trainning dan testing
3. Tampilan data hasil klasifikasi, yang akan menunjukkan suatu dokumen akan termasuk kedalam kelas yang mana.
4. Data file teks yang menyimpan hasil perhitungan presentase rata – rata keakuratan aplikasi mengenali kelas dokumen,

4.1.2 PERANCANGAN PROSES

Untuk melakukan perancangan proses dalam tugas akhir ini, penulis menggunakan teknik *functional based model* atau yang lebih sering dikenal dengan Data Flow Diagram (DFD). Sedangkan perangkat lunak yang digunakan untuk menggambar diagram DFD adalah Power Designer 6 – process analyst.

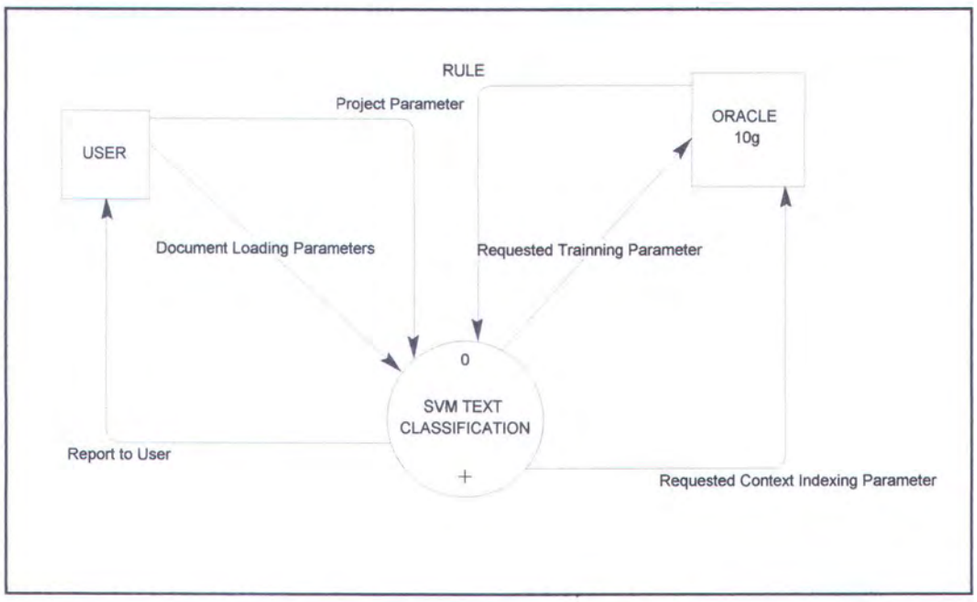
Secara umum DFD yang dibuat dapat disimpulkan ke dalam bentuk Process Tree pada gambar 4.2,



Gambar 4.2 Process Tree

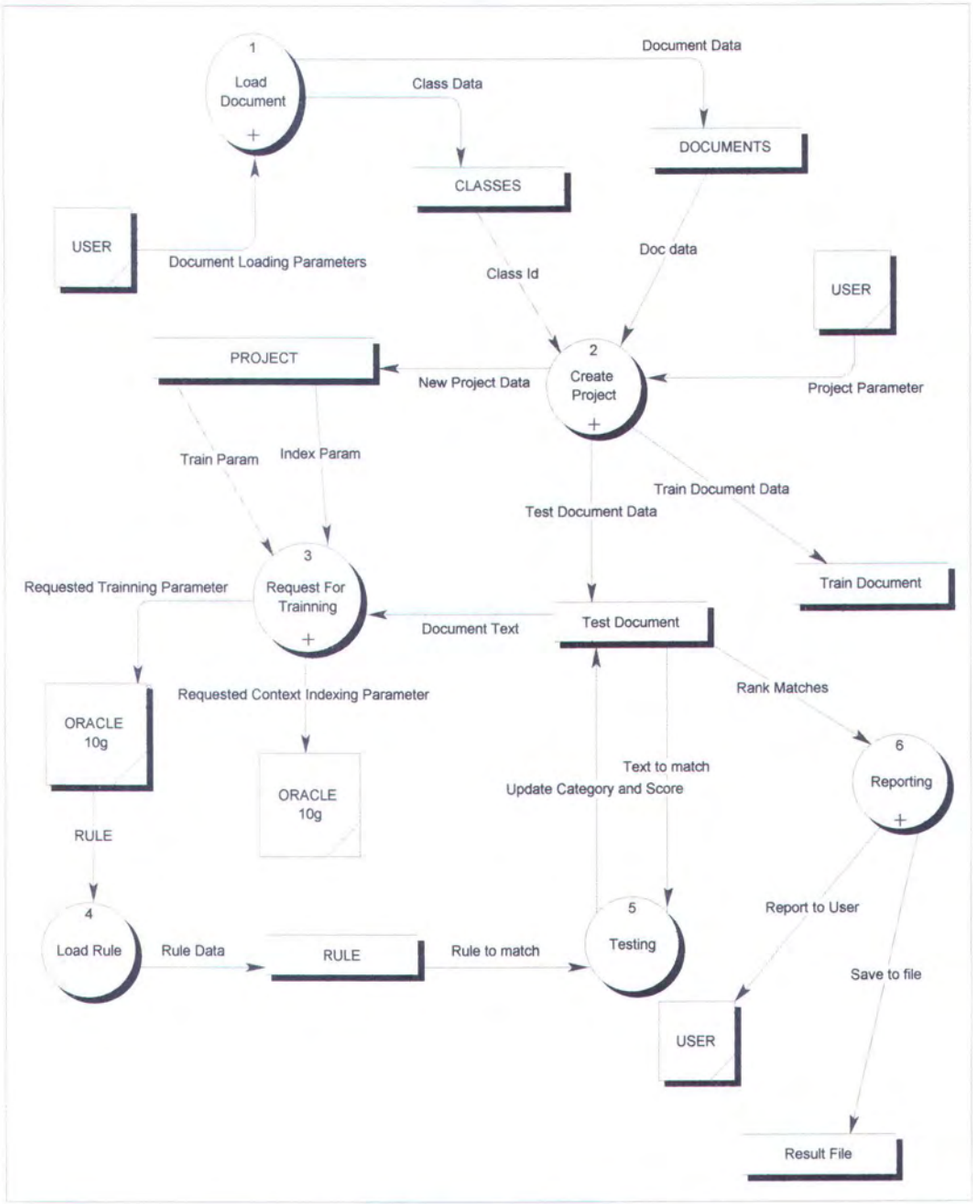
Setiap penggambaran DFD akan dimulai dari penggambaran context diagramnya, yang dapat dilihat pada gambar dibawah ini. Context diagram ini dapat juga diartikan sebagai batasan dari sistem / aplikasi perangkat lunak yang akan dibuat. Pada diagram tersebut terdapat 2 entitas luar, yaitu pengguna aplikasi dan oracle 10g dalam hal ini adalah *Oracle Text engine*. Entitas user bertanggung jawab untuk memasukkan data masukkan yang tergambar sebagai 2 data flow yaitu : *Project parameter*, dan *Document Loading Parameter*. Oracle 10g dalam hal ini digambarkan sebagai entitas luar yang akan berfungsi sebagai engine

dalam melakukan trainning klasifikasi dan indeks. Oracle akan mengembalikan Rule kedalam sistem sebagai hasil dari proses trainningnya.



Gambar 4.3 Context Diagram (DFD level 0)

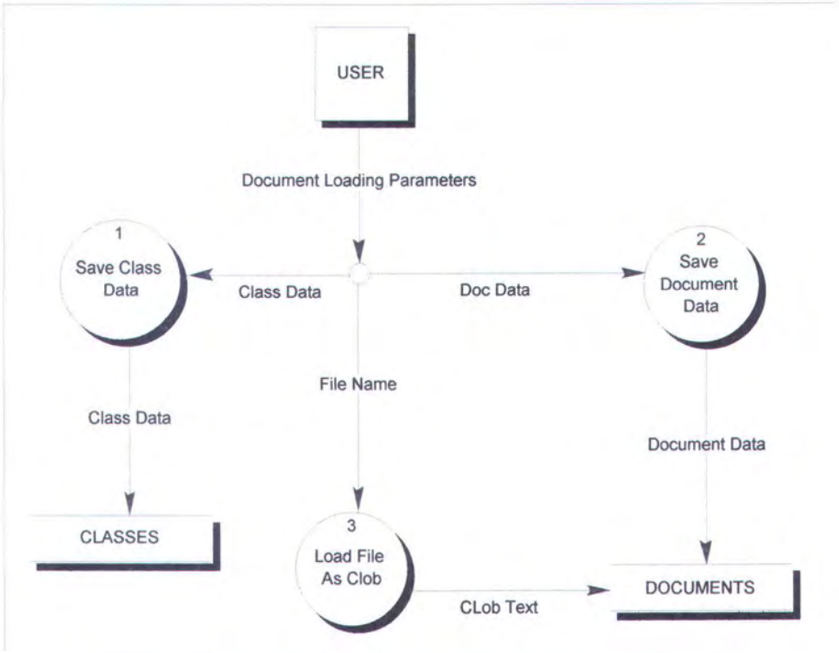
Diagram context diatas di jelaskan dengan dekomposisi nya pada DFD level 1, atau yang lebih sering di sebut sebagai diagram overview sebagaimana yang terdapat pada gambar 4.4. Pada overview diagram ini akan digambarkan proses – proses utama yang dijalankan oleh sistem. Di dalam *overview diagram* terdapat 6 buah proses utama, yaitu : load document, create project, trainning (request for trainning), load rule, testing dan reporting.



Gambar 4.4 Overview Diagram (DFD level 1)

4.1.2.1 PROSES LOAD DOCUMENT

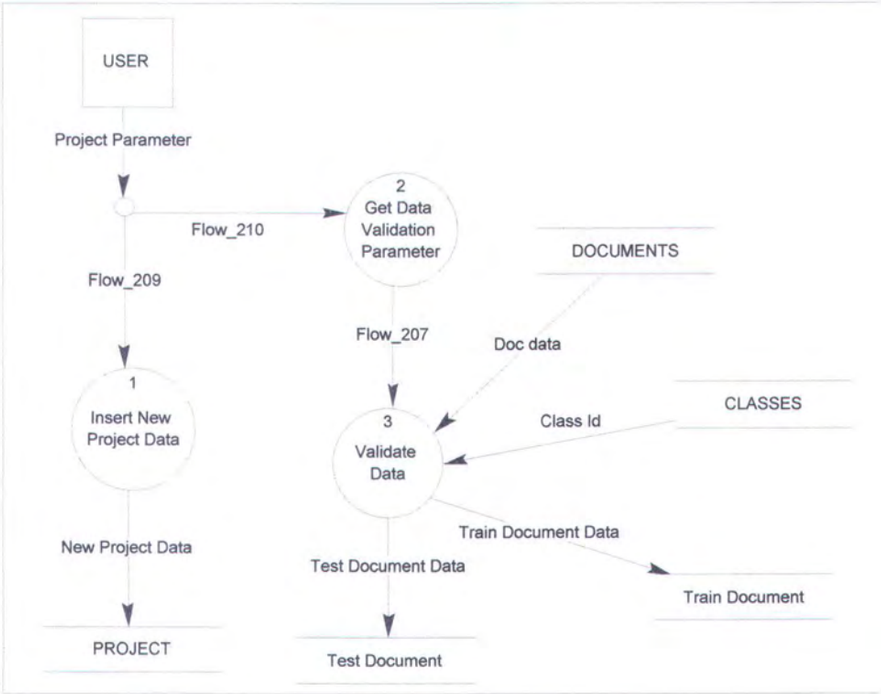
Diagram dibawah ini menggambarkan dekomposisi dari proses pertama yang ada pada diagram overview.



Gambar 4.5 DFD level 2 untuk proses load document

4.1.2.2 PROSES CREATE PROJECT

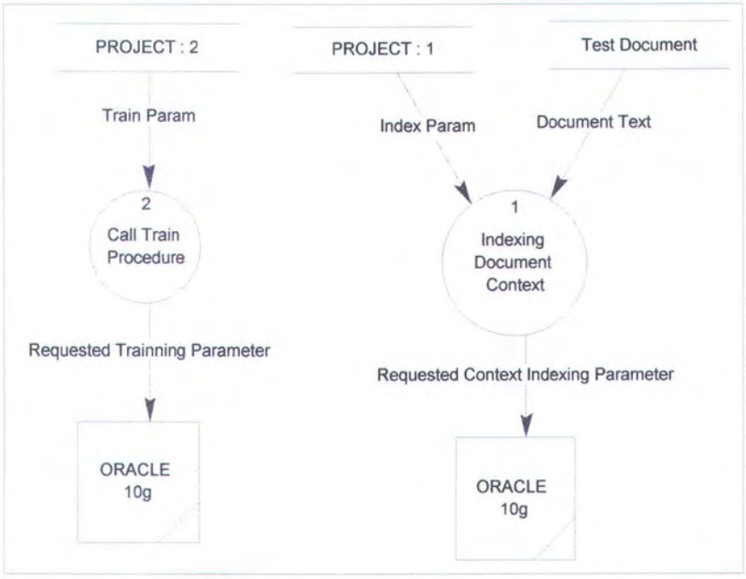
Proses create project merupakan proses untuk user memasukkan data – data atau atribut proses, dimana untuk kepentingan analisa kamampuan, disediakan mekanisme pergantian data atribut dengan menggabungkan satu paket atribut (training dan index) kedalam satu paket yang disebut sebagai project. Sehingga diharapkan pada saat aplikasi berjalan user dapat membuat beberapa project sekaligus atau membuka kembali setting aplikasi yang sudah pernah dilakukan sebelumnya.



Gambar 4.6 DFD level 2 untuk proses create project

4.1.2.3 PROSES TRAINNING

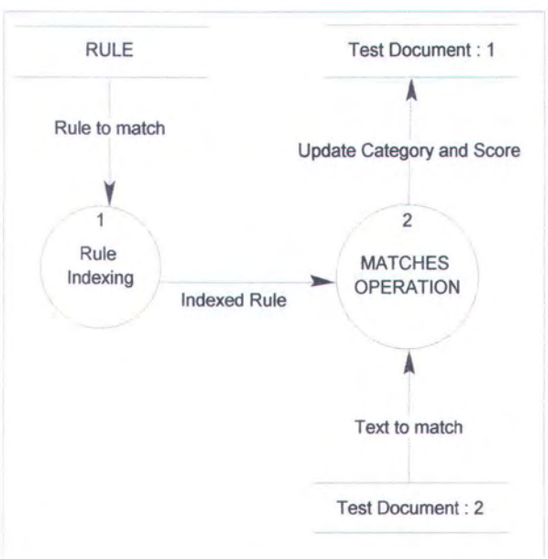
Proses training dalam desain proses dijelaskan sebagai proses yang terdiri atas 2 sub proses dibawahnya yaitu, proses *indexing* teks dari dokumen yang akan diklasifikasi dengan parameter indeksnya. Proses berikutnya dalah pemanggilan prosedur `CTX_CLS.TRAIN` yang terdapat pada oracle text. Pemanggilan prosedur ini juga disertai dengan pengiriman parameter training.



Gambar 4.7 DFD level 2 untuk proses Trainning

4.1.2.4 PROSES TESTING

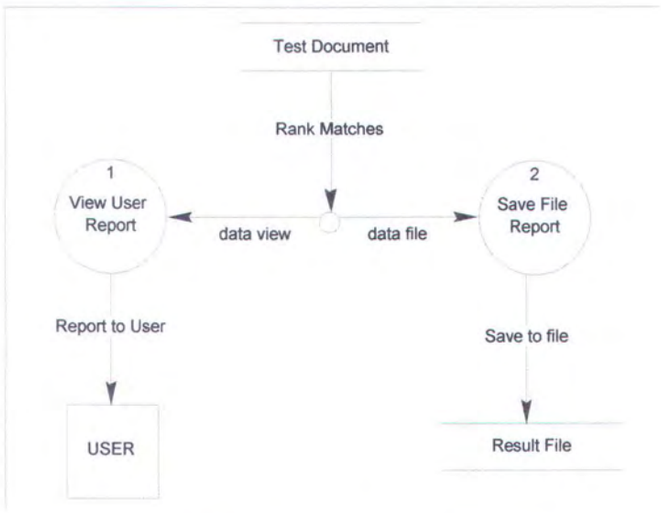
Proses testing adalah proses pengujian rule yang dihasilkan oleh proses trainning, dan rule tersebut harus sudah di indeks terlebih dahulu. Proses ini dilakukan dengan membadingkan text dokumen dengan rule yang telah di indeks.



Gambar 4.8 DFD level 2 untuk proses Testing

4.1.2.5 PROSES REPORTING

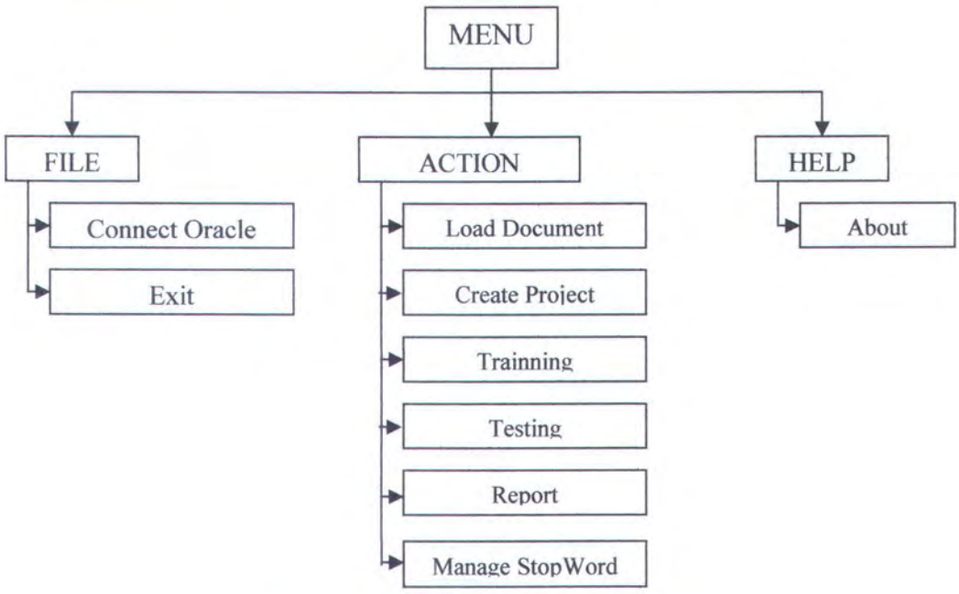
Diagram dibawah ini merupakan dekomposisi dari proses reporting, yang terdiri atas 2 sub proses, yaitu proses view user report dan proses save file report, view user report akan menampilkan data –data report ke user dalam bentuk tampilan pada aplikasi. Data – data ini sesuai dengan data keluaran yang telah di bahas pada bab tentang desain data keluaran. Sedangkan data keluaran lainnya adalah dalam bentuk file teks yang memiliki format *.csv (file yang berisi serangkaian teks yang dipisahkan dengan tanda koma).



Gambar 4.9 DFD level 2 untuk proses Reporting

4.1.3 PERANCANGAN ANTAR MUKA

Perancangan antar muka ini dimaksudkan untuk mencapai kemudahan pemakaian bagi pengguna. Juga untuk menjelaskan bagaimana fasilitas – fasilitas yang disediakan dalam perangkat lunak ini diatur dalam susunan menu dan kontrol pengguna. Struktur menu yang disediakan dalam aplikasi ini dibagi menjadi 3 bagian menu utama, yaitu menu File, menu Action, dan Menu Help. Secara lengkap digambarkan dalam gambar hirarki berikut ini :



Gambar 4.10 Hirarki menu aplikasi

Menu File berisi sub menu connect Oracle dan Exit. Fasilitas *Connect Oracle* digunakan untuk melakukan koneksi dengan data server oracle. Sedangkan sub menu Exit digunakan untuk keluar dari aplikasi.

Menu Action terdiri atas aktifitas – aktifitas utama yang dapat dilakukan dengan aplikasi ini, yang terdiri atas sub menu *Load Document*, fasililtas ini digunakan untuk memasukkan file – file dokumen ke dalam data server, file – file ini nantinya yang akan dipopulasi ke dalam kelas – kelas, baik sebagai data

training mapun sebagai data testing. Sub Menu *Create Project* memungkinkan user untuk melakukan serangkaian percobaan terhadap hasil training dan testing aplikasi, karena satu project berisi sekumpulan nilai – nilai parameter training dan indeks. Sub menu *Testing* digunakan untuk melakukan proses testing terhadap rule – rule yang dihasilkan oleh proses trainning. Sedangkan sub menu *Report* digunakan untuk menampilkan serangkaian laporan, dengan membaca hasil testing, dalam beberapa bentuk laporan. Menu *Manage Stopword* digunakan untuk mendefinisikan file yang berisi *stop list*, yaitu kata – kata yang tidak perlu dianggap sebagai token dalam proses trainning.

4.2 IMPLEMENTASI PERANGKAT LUNAK

Setelah tahap perancangan telah diselesaikan, maka desain perangkat lunak sudah dapat diimplementasikan. Tahap – tahap implementasi mengikuti tahapan peracangannya, yaitu implementasi untuk rancangan data, proses dan antar muka.

4.2.1 LINGKUNGAN IMPLEMENTASI

Perangkat lunak ini akan diimplementasikan dalam lingkungan sebagai berikut :

1. Perangkat keras

   Aplikasi ini nantinya akan berjalan pada sebuah *personal computer* dengan processor AMD Ahlon 2 GHz dan memori (RAM) 1 Giga Byte.

2. Perangkat lunak

   - Sistem Operasi, perangkat lunak ini akan dikembangkan pada sistem operasi Windows2000 Server

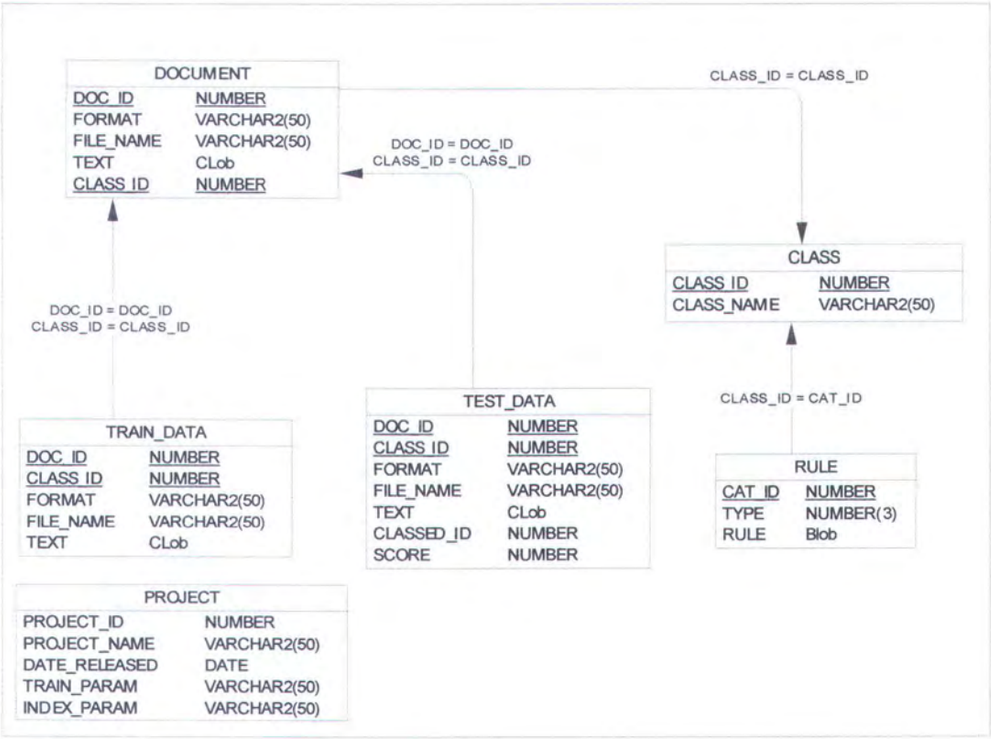
- Server Basis data, manajemen data pada perangkat lunak ini akan diaplikasikan menggunakan Basis data Oracle 10g beserta perangkat Oracle Text 10.1.0.2.0
- Bahasa pemrograman yang digunakan untuk membangun perangkat lunak ini adalah bahasa pemrograman Java, dengan IDE NetBeans 4.0

4.2.2 IMPLEMENTASI DATA

Data yang diimplementasikan terdiri atas data masukan, data proses dan data keluaran.

4.2.2.1 IMPLEMENTASI DATA MASUKAN

Data trainning dan Testing disimpan dalam Basis data Oracle 10g. Sedangkan Model Fisik dari basis data yang akan digunakan tampak apda gambar 4.11 berikut ini.



Gambar 4.11 Model fisik basis data

Tabel *Document* menyimpan data teks dari dokumen beserta atribut lainnya, dimana setiap *record*-nya diasosiasikan dengan satu file fisik yang ada di komputer. Tabel ini memilki atribut *Doc_id* sebagai *primary key*, atribut *file name* untuk menyimpan nama file fisik dari teks dokumen. Sedangkan atribut *text* yang bertipe *Clob* digunakan untuk menyimpan teks dari dokumen. Atribut *format* digunakan untuk menyimpan data jenis format file dari teks dokumen yang akan disimpan. Atribut *class id* ditambahkan karena untuk menjalankan proses *supervised trainning* dokumen - dokumen yang akan di *trainning* harus sudah di klasifikasikan terlebih dahulu.

Tabel *Class* digunakan untuk menyimpan data kelas yang digunakan untuk mengklasifikasikan dokumen, tabel ini memiliki *primary key* yaitu atribut *class_id*, dan atribut *class_name* untuk menyimpan nama dari tiap kelas.

Tabel *Project* digunakan untuk menyimpan parameter – parameter dari *project*, seperti yang telah dijelaskan sebelumnya, tabel ini dapat menelusuri history penggunaan aplikasi, terutama data – data parameter indeks dan trainning yang dipilih oleh pengguna. Tabel ini memiliki atribut *Project_id* sebagai *primary key*, atribut *project_name* untuk mendefinisikan nama dari setiap *project*, atribut Date_released digunakan untuk menyimpan data tanggal dan waktu saat project dirilis, atribut *index_param* untuk menyimpan data – data atribut indeks, dan *train_param* digunakan untuk menyimpan data atribut trainning. Atribut – atribut ini disimpan dalam format string dengan delimiter berupa '/' yang memisahkan setiap atribut dengan atribut lainnya.

Ketiga tabel lainnya yaitu *Test_Data*, *Train_Data* dan *Rule* pada gambar hanya akan bersifat sebagai *template*, dengan maksud setiap kali project baru dibuat, maka ketiga tabel tersebut akan dibuat ulang dengan aturan nama : `nama_project + nama_template`, misal pada saat pengguna membuat *project* baru dengan nama prj1 maka akan dibuat tabel test_data dengan nama prj1test_data. Hal ini bertujuan agar setiap kali data disimpan dan harus diindeks prosesnya tidak akan memakan waktu lama. Dan setiap *project* akan memiliki ketiga tabel tersebut sebagai data historinya. Tabel *Train_Data* digunakan untuk menyimpan data – data yang akan digunakan sebagai masukan pada proses trainning, data – data tersebut diambil dari tabel *document* dengan sebelumnya user harus menentukan terlebih dahulu aturan validasi yang akan dipakai (3-fold atau 10-fold). Atribut – atribut pada tabel ini sama dengan tabel *Document*. Sedangkan table *Test_Data* akan berisi data – data dokumen yang akan digunakan sebagai input dalam proses *testing*. Atribut – atributnya sama dengan tabel Train_data dengan penambahan 2 atribut lainnya yaitu : *classed_id* dan *score*, atribut *classed_id* digunakan untuk menyimpan id kelas hasil pengenalan oleh proses *testing*, dan score pengenalannya akan disimpan dalam kolom *score*. Tabel *Rule* digunakan untuk menyimpan aturan – aturan yang dihasilkan oleh proses *trainning* dokumen. Tabel ini memiliki atribut *cat_id* digunakan untuk menyimpan id kelas yang berhubungan dengan suatu aturan, *type* digunakan untuk menyimpan jenis hubungan (karena dimungkinkan muncul kelas id baru, maka atribut ini akan bernilai 1 apabila yang disimpan adalah id kelas yang asli, dan 0

apabila yang disimpan adalah id kelas tambahan). Sedangkan atribut *rule* digunakan untuk menyimpan aturan dari setiap kelas.

4.2.2.2 IMPLEMENTASI DATA PROSES

Sesuai dengan perancangan, data-data utama pada proses dibagi menjadi dua berdasarkan jenis prosesnya :

1. Indexing

Data ini diimplementasikan pada kelas FrIndex dengan penginisialisasian preference stemming, stopwords, dan penggunaan thesaurus yang dipakai sesuai pilihan pengguna dengan prosedur CTX_DDL.CREATE_PREFERENCE.

Proses penginisialisasian penggunaan proses stemming dengan CTX_DDL.CREATE_PREFERENCE ini dapat dilihat dibawah ini:

```
begin
ctx_ddl.create_preference('MY_WORDLIST', 'BASIC_WORDLIST');
ctx_ddl.set_attribute('MY_WORDLIST','STEMMER','DERIVATIONAL');
end;
```

Untuk proses penginisialisasian penggunaan penghilangan stopwords untuk proses indexing dengan CTX_DDL.CREATE_STOPLIST ini dapat dilihat dibawah ini:

```
Begin
ctx_ddl.create_stoplist('MY_STOPLIST','BASIC_STOPLIST');
end;
```

Untuk proses penginisialisasian penggunaan thesaurus untuk proses indexing dengan CTX_DDL.SET_ATTRIBUTE ini dapat dilihat dibawah ini:

```
BEGIN
ctx_ddl.set_attribute('MY_LEXER','index_themes','YES');
ctx_ddl.set_attribute('MY_LEXER','theme_language','ENGLISH');
END;
```

2. Validation

   Pada Validation diperlukan data hasil clustering dan masukan berupa penggunaan 3 fold validation atau 10 fold validation.

3. Training

   Pada Training diperlukan data hasil Validation dan masukan berupa nilai max term, threshold, NT threshold, dan Term threshold yang diimplementasikan pada kelas FrTrainClass. Untuk menginisialisasikan parameter tersebut, digunakan prosedur CTX_DDL.CREATE_PREFERENCE seperti dibawah ini:

```
1  BEGIN
2  ctx_ddl.create_preference('my_classifier','SVM_CLASSIFIER');
3  ctx_ddl.set_attribute('my_classifier','MAX_DOCTERMS','20');
4  ctx_ddl.set_attribute('my_classifier','MAX_FEATURES','40');
5  ctx_ddl.set_attribute('my_classifier','THEME_ON','TRUE');
6  ctx_ddl.set_attribute('my_classifier','TOKEN_ON','TRUE');
7  ctx_ddl.set_attribute('my_classifier','STEM_ON','TRUE');
8  END;
```

4.2.2.3 IMPLEMENTASI DATA KELUARAN

Data – data yang dihasilkan dalam aplikasi ini terdiri atas :

1. Daftar dokumen yang digunakan dalam satu kali trainning

   Data ini berupa tampilan pada antar muka pengguna, yang berupa tampilan tabel yang terdiri atas kolom – kolom antara lain : Id dokumen, Kelas Dokumen , Nama File Dokumen, Format File Dokumen dan Ukuran File Dokumen.

2. Daftar dokumen yang digunakan dalam testing beserta hasil klasifikasi yang dilakukan oleh sistem.

   Data ini juga berupa tampilan tabel pada antar muka pengguna, yang terdiri atas kolom – kolom : Id dokumen, Kelas Dokumen , Nama File Dokumen, Kelas hasil pengenalan aplikasi,. dan score klasifikasinya.

3. Data laporan hasil testing, data ini dibentuk dalam sebuah file teks, dengan format *.CSV (file teks dengan *comma delimmiter*), file ini memiliki atribut kelas id, dan presentase keberhasilkan pengenalan kelas dokumen dalam satu kelas, yang diujicobakan pada data testing.

4.2.3 IMPLEMENTASI PROSES

Sesuai dengan desain proses yang sudah dijelaskan pada sub bab sebelumnya, maka pada sub bab ini penjelasan implementasi proses dibagi menjadi : implementasi indexing , implementasi trainning, dan implementasi testing, sementara proses yang lain tidak akan dibahas dalam sub bab ini, dikarenakan dalam pembahasan klasifikasi teks, tiga proses tersebutlah yang akan menjadi perhatian.

4.2.3.1 IMPLEMENTASI PROSES INDEXING

Implementasi proses indexing pada desain proses, adalah proses untuk membaca data setting indeks yang berasal dari masukan pengguna, dan kemudian mengaplikasikannya untuk mengatur *index preference*. Proses indexing ini nantinya akan digunakan pada saat proses trainning, untuk mengindeks teks dokumen test.

4.2.3.2 IMPLEMENTASI PROSES TRAINNING

Berikut ini merupakan prosedur `CTX_CLS.TRAIN` yang digunakan untuk melakukan proses trainning.

```
BEGIN
CTX_DDL.SYNC_INDEX('TRAINTABLE_INDEX','2M');
ctx_cls.train('TRAINTABLE_INDEX','doc_id','TRAINTABLE',
'doc_id','class_id','TABEL_RULE','my_classifier');
END;
```

4.2.3.3 IMPLEMENTASI PROSES TESTING

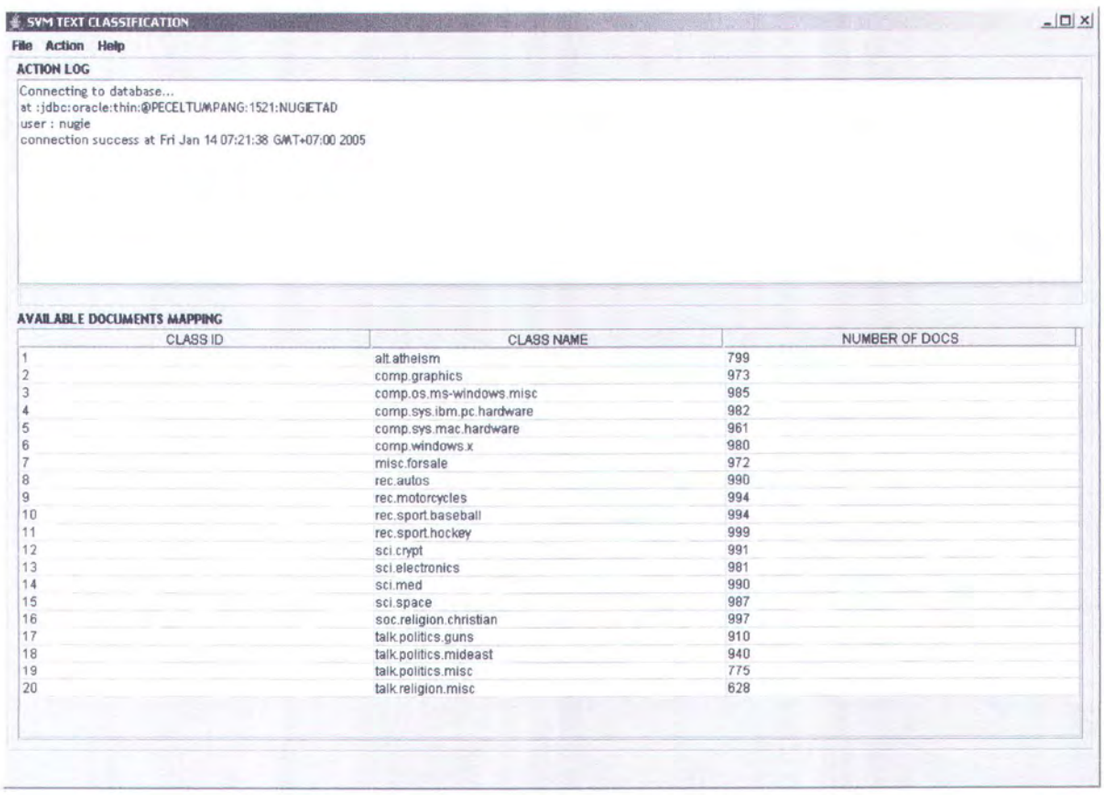
Proses testing dalam aplikasi ini akan menjalankan operasi pengujian rule yang dihasilkan oleh proses trainning, dengan dokumen – dokumen yang digunakan sebagai data testing. Berikut ini adalah sintaks *query* yang digunakan untuk melakukan pengujian tersebut. Dalam sintaks *qeury* ini dijalankan tiga lapis query, lapis terdalam adalah untuk memeriksa teks dokumen yang dimasukkan sebagai parameter query (disimbolkan dengan tanda '?') akan dikenali dalam kelas apa saja. Karena dimungkinkan suatu dokumen akan di menghasilkan nilai *match score* yang lebih besar dari 0, untuk beberapa kelas, maka perlu dilakukan

query sekali lagi untuk menjalankan proses ranking, dan mencari score terbesar yang dimiliki oleh hasil klasifikasi terhadap suatu dokumen.

```
select * from
(select a.cat_id,sc,rank() over(order by a.sc desc) rank from
(select cat_id, match_score(1) as sc from tabel_rule where
 matches(rule,?,1)>0 order by match_score(1)) a) where rank=1;
```

4.2.4 IMPLEMENTASI ANTAR MUKA PENGGUNA

Pada sub bab ini akan dijelaskan mengenai implementasi antar muka pengguna, pada gambar 4.12 berikut, ditampilkan form utama aplikasi.



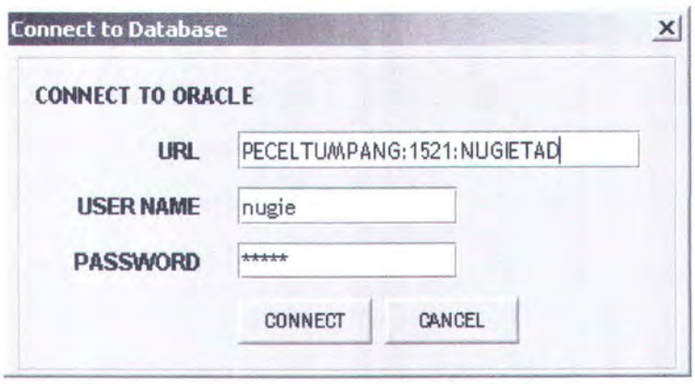
CLASS ID	CLASS NAME	NUMBER OF DOCS
1	alt.atheism	799
2	comp.graphics	973
3	comp.os.ms-windows.misc	985
4	comp.sys.ibm.pc.hardware	982
5	comp.sys.mac.hardware	961
6	comp.windows.x	980
7	misc.forsale	972
8	rec.autos	990
9	rec.motorcycles	994
10	rec.sport.baseball	994
11	rec.sport.hockey	999
12	sci.crypt	991
13	sci.electronics	981
14	sci.med	990
15	sci.space	987
16	soc.religion.christian	997
17	talk.politics.guns	910
18	talk.politics.mideast	940
19	talk.politics.misc	775
20	talk.religion.misc	628

Gambar 4.12 FormUtama dari aplikasi

Seperti yang telah dibahas pada bab perancangan, strukur menu yang terdapat pada form utama ini antara lain :

- Menu File

Menu ini hanya digunakan untuk mengatur koneksi dengan Database Server Oracle 10g. Dengan terlebih dahulu memilih sub menu *Connect Database* dan kemudian akan tampil, form koneksi database seperti yang tampak pada gambar 4.13 berikut ini. Untuk melakukan koneksi ke database, pengguna harus memasukkan URL , yang terdiri atas 3 bagian yaitu : nama server, port koneksi dan nama service. Selain ituuser juga harus memasukkan user name database beserta passwordnya.

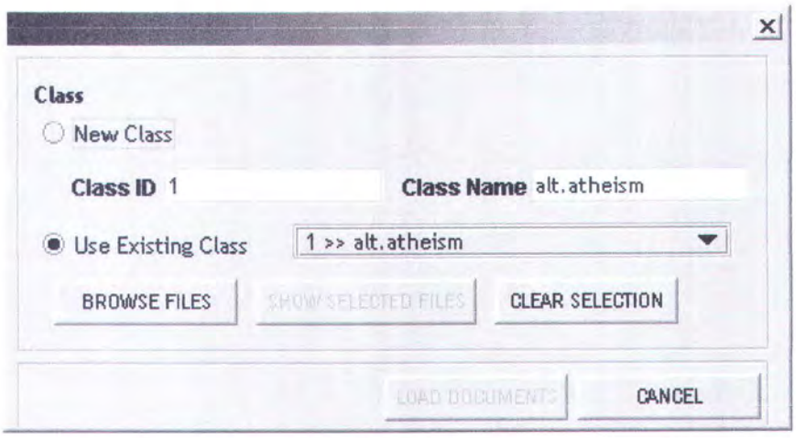


Gambar 4.13Form Koneksi Ke database

- Menu Action

Menu Action ini berisi sebagian besar sub menu yang ada pada aplikasi, yang terdiri atas : sub menu load document, Open Project, New project, Trainning, Testing, Project properties, dan set stop word. Sub menu load document digunakan untuk memanggil form load document seperti yang tampak pada gambar 4.14 di bawah ini. Form ini dapat digunakan untuk memasukkan dokumen – dokumen kedalam database dan sekaligus mempopulasikannya dengan label kelas terterntu. Untuk membuat kelas baru user harus memilih radio button *new class* dan kemudian memasukkan

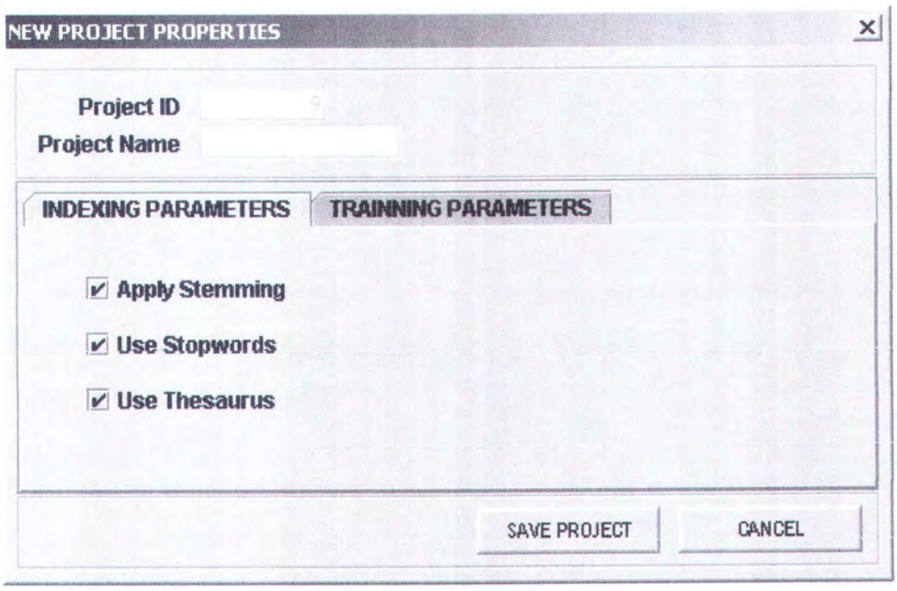
Nama Kelas. User juga dapat menggunakan nama kelas yang sudah ada dengan memilih dari *combo box* kelas, dengan terlebih dahulu mengaktifkan radio button *Use existing class.* Untuk melakukan loading document user harus menekan tombol *browse file*, untuk menentukan file – file mana yang akan di-load. Setelah penentuan file, tombol *load document* harus ditekan.



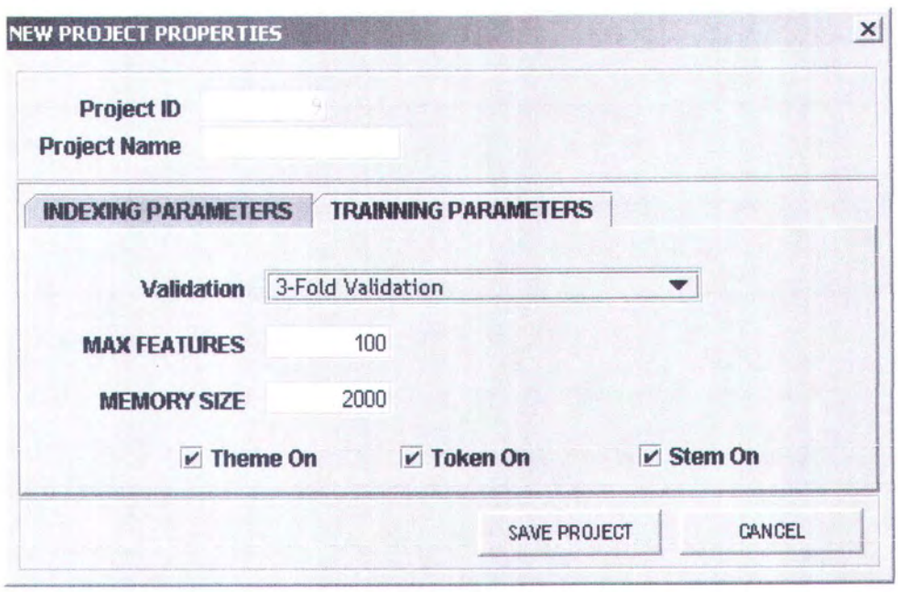
Gambar 4.14 Form Loading Document

Pada menu action ini juga user dapat mengatur jalannya training dan testing klasifikasi dokumen, dengan terlebih dahulu user harus memilih sub menu *new project*, seperti pada penjelasan rancangan proses, project adalah istilah yang digunakan untuk mengatur setting parameter indeks dan trainning aplikasi. Seperti pada gambar 4.15. Untuk membuat *project* baru, pengguna harus memasukkan project name, serta memenuhi pilihan parameter indeks dan parameter trainning (untuk parameter trainning dapat di lihat pada gambar 4.16).

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER



Gambar 4.15 Form Create New Project (index setting)



Gambar 4.16 Form Create New Project (index setting)

Berikutnya setelah project dibuat, proses trainning dapat dilakukan dengan memilih sub menu *trainning*, yang kemudian juga dapat

dilanjutkan dengan proses *testing* terhadap rule hasil *trainning*.Pada saat proses testing berkahir, user akan secara otomatis di minta untuk menentukan nama file beserta lokasi penyimpanannya untuk menampung data hasil analisa testing.

- Menu Help

Menu ini hanya terdiri atas menu *about* saja.

Setiap aktifitas yang dilakukan oleh penggguna akan di tampilkan ke dalam action log yang terdapat pada menu utama, hal ini bertujuan agar pengguna selalu dapat melacak aktifitasnya terhadap pemakaian aplikasi.

BAB V

UJI COBA DAN EVALUASI

BAB V

UJICOBA DAN EVALUASI

UJICOBA DAN EVALUASI

Uji coba terhadap aplikasi text classification ini dilakukan terhadap data UseNet Collection (20 NG) yang terdiri dari 20 newsgropus yang berisi hampir 1000 dokumen, setiap kelasnya.

Uji coba ini juga dibandingkan dengan data hasil aplikasi klasifikasi teks berbasis decision tree.

5.1 LINGKUNGAN PENGUJIAN

Uji coba dilakukan pada komputer dengan spesifikasi perangkat keas sebagai berikut : processor AMD Athlon 2 GHz, dan memori (RAM) 1 Gb. Sistem operasi yang digunakan adalah Microsoft Windows 2000 Server dengan basis data oracle 10g dan Oracle Text 10.1.

5.2 DATA UJI COBA

Data untuk uji coba perangkat lunak ini adalah data set dari UseNet Collection (20NG) yang dikumpulkan oleh Ken Lang. Jumlah dokumen yang tersimpan dalam data set ini hampir mencapai 20.000 dokumen newsgroup. Dokumen – dokumen tersebut telah diklasifikasikan kedalam 20 newsgroups yang berbeda.

UseNet Collection terdiri dari 20,000 newsgroup yang dikumpulkan dari 20 newsgroup yang berbeda di tahun 1992 (1000 dokumen per newsgroup). Data tersebut diorganisasikan menjadi 20 newgroup yang berbeda, tiap newgroup tersebut terkorespondensi ke suatu topik yang berbeda. Beberapa newgroup

sebenarnya sangat berhubungan dekat, misalnya comp.sys.mac.pc.hardware dan comp.sys.mac.hardware, sedangkan yang lainnya sangat tidak berhubungan, seperti misc.forsale dan soc.religion.christian. Berikut ini merupakan list dari 20 newsgroup tersebut, yang dipartisi (kurang lebih) menurut subjek masalahnya :

Tabel 5.1 Data UseNet Collection menurut subyek

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

Adanya variasi dari dataset membuat hanya 18,941 berita posting yang diambil, karena posting yang duplikat dihilangkan untuk menghindari perulangan pada contoh training. Keduapuluh newsgroup tersebut adalah:

Tabel 5.2 Deskripsi data set UseNet Collection

Kelas	**Total Data**
alt.atheism	799 file
comp.graphics	979 file
comp.os.ms-windows.misc	989 file
comp.sys.ibm.pc.hardware	984 file
comp.sys.mac.hardware	989 file
comp.windows.x	992 file
Misc.forsale	985 file
rec.autos	991 file
rec.motorcycles	999 file
rec.sport.baseball	1000 file
rec.sport.hockey	1000 file

sci.crypt	999 file
sci.electronics	987 file
sci.med	993 file
sci.space	990 file
soc.religion.christian	997 file
talk.politics.guns	915 file
talk.politics.mideast	948 file
talk.politics.misc	777 file
talk.religion.misc	628 file

5.3 PELAKSANAAN UJI COBA

Uji coba perangkat lunak dilaksanakan dengan 2 tujuan utama, yaitu :

1. Membadingkan hasil uji coba dengan hasil yang diperoleh pada decision tree classifier yang dibangun oleh Nuning [Pur-2004], aplikasi tersebut juga melakukan uji coba data pada data UseNet Collection (20NG). Mengingat *decision tree classifier* juga termasuk sebagai metode *supervised*, maka perbandingan ini bisa dilakukan. Dengan harapan dapat membuktikan hipotesa bahwa *SVM classifier* memiliki kemampuan yang lebih baik dalam mengklasifikasikan dokumen.
2. Menguji pengaruh perubahan parameter indeks terhadap kemampuan klasifikasi dokumen.

5.3.1 PERBANDINGAN UJI COBA

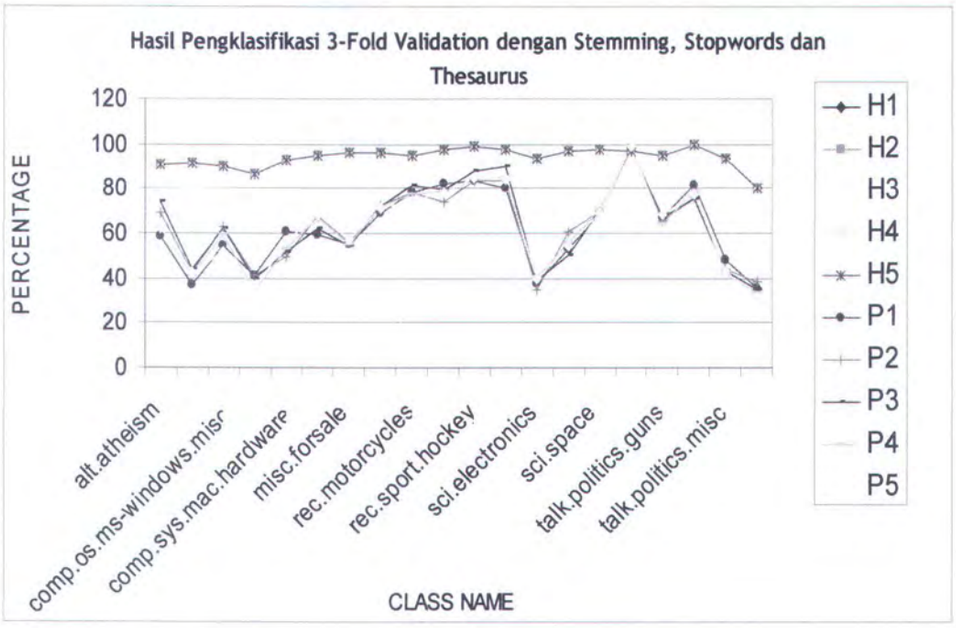
Berikut disajikan hasil uji coba perangkat lunak, dan sekaligus membandingkan dengan hasil yang diperoleh dari Decision Tree Classifier.

5.3.1.1 UJI COBA DENGAN 3-FOLD DATA VALIDATION

- **Uji coba I dengan penggunaan *Stemming*, *Thesurus dan Penghilangan Stopwords***

Uji coba pertama dilakukan dengan menentukan parameter indeks secara lengkap, yaitu penggunaan stemming, dan penghilangan stopword. Sedangkan untuk parameter trainning di set secara default.

Uji coba dilakukan sebanyak lima kali (H1 sampai dengan H5), serta dihitung pula kemampuan rata – rata dari lima kali uji coba tersebut. Hasil uji coba ini kemudian dibandingkan dengan data uji coba pada *decision tree classifier* (P1 sampai dengan P5) , dapat dilihat pada gambar 5.1



Gambar 5.1 Hasil uji coba dan perbandingan klasifikasi data dengan 3-fold validation, apply stemming, penghilangan stopwords dan penggunaan thesaurus

Pada gambar 5.1, grafik H1 sampai dengan H5 adalah garifk yang menunjukkan hasil klasifikasi dengan SVM classifier, sedangkan grafik dengan label P1 sampai dengan P5 adalah grafik kemampuan klasifikasi yang dimiliki oleh decision tree classiier.

Berikut ini juga ditampilkan tabel numerik yang menunjukkan angka – angka presentase yang lebih mendetail.

Tabel 5.3 Tabel perbandingan rata- rata kemampuan klasifikasi antara Dec. Tree classifier dengan SVM classifier, dengan 3-fold data validation, penggunaan stemming, penghilangan stopword dan penggunaan thesaurus

Pesentase kemampuan rata – rata klasifikasi dokumen pada SVM Classifier							
	CLASS NAME	**H1**	**H2**	**H3**	**H4**	**H5**	**Rata - rata**
1	alt.atheism	91.01%	91.01%	91.01%	91.01%	91.01%	91.01%
2	comp.graphics	91.69%	91.69%	91.69%	91.69%	91.69%	91.69%
3	comp.os.ms-windows.misc	89.67%	89.67%	89.67%	89.67%	89.67%	89.67%
4	comp.sys.ibm.pc.hardware	86.28%	86.28%	86.28%	86.28%	86.28%	86.28%
5	comp.sys.mac.hardware	92.83%	92.83%	92.83%	92.83%	92.83%	92.83%
6	comp.windows.x	95.12%	95.12%	95.12%	95.12%	95.12%	95.12%
7	misc.forsale	96.60%	96.60%	96.60%	96.60%	96.60%	96.60%
8	rec.autos	96.06%	96.06%	96.06%	96.06%	96.06%	96.06%
9	rec.motorcycles	94.88%	94.88%	94.88%	94.88%	94.88%	94.88%
10	rec.sport.baseball	97.89%	97.89%	97.89%	97.89%	97.89%	97.89%
11	rec.sport.hockey	99.40%	99.40%	99.40%	99.40%	99.40%	99.40%
12	sci.crypt	97.89%	97.89%	97.89%	97.89%	97.89%	97.89%
13	sci.electronics	93.58%	93.58%	93.58%	93.58%	93.58%	93.58%
14	sci.med	96.97%	96.97%	96.97%	96.97%	96.97%	96.97%
15	sci.space	97.87%	97.87%	97.87%	97.87%	97.87%	97.87%
16	soc.religion.christian	97.30%	97.30%	97.30%	97.30%	97.30%	97.30%
17	talk.politics.guns	95.07%	95.07%	95.07%	95.07%	95.07%	95.07%
18	talk.politics.mideast	99.68%	99.68%	99.68%	99.68%	99.68%	99.68%
19	talk.politics.misc	93.82%	93.82%	93.82%	93.82%	93.82%	93.82%
20	talk.religion.misc	80.48%	80.48%	80.48%	80.48%	80.48%	80.48%
	Kemampuan rata - rata	**94.20%**	**94.20%**	**94.20%**	**94.20%**	**94.20%**	**94.20%**
Pesentase kemampuan rata – rata klasifikasi dokumen pada Dec. Tree Classifier							
	CLASS NAME	**P1**	**P2**	**P3**	**P4**	**P5**	**Rata - rata**
1	alt.atheism	58.80%	68.91%	74.16%	61.80%	61.80%	65.09%
2	comp.graphics	37.00%	42.81%	44.04%	40.98%	42.51%	41.47%
3	comp.os.ms-windows.misc	54.08%	62.84%	61.93%	59.21%	51.36%	57.88%
4	comp.sys.ibm.pc.hardware	41.46%	40.85%	39.63%	33.84%	34.76%	38.11%
5	comp.sys.mac.hardware	61.03%	49.85%	51.66%	52.87%	56.50%	54.38%
6	comp.windows.x	59.04%	67.47%	62.05%	66.87%	68.37%	64.76%
7	misc.forsale	55.02%	55.93%	54.10%	55.32%	57.45%	55.56%
8	rec.autos	68.88%	72.81%	71.90%	71.60%	71.30%	71.30%
9	rec.motorcycles	78.08%	78.08%	81.68%	77.78%	74.77%	78.08%
10	rec.sport.baseball	82.63%	74.25%	79.64%	79.04%	78.44%	78.80%
11	rec.sport.hockey	83.53%	83.83%	88.02%	84.73%	85.33%	85.09%
12	sci.crypt	80.18%	83.48%	89.79%	84.98%	84.08%	84.50%
13	sci.electronics	37.99%	34.95%	39.21%	40.12%	39.82%	38.42%
14	sci.med	52.87%	60.73%	50.45%	55.89%	54.68%	54.92%

15	sci.space	68.79%	70.00%	70.61%	70.30%	69.39%	69.82%
16	soc.religion.christian	98.50%	99.10%	98.50%	98.80%	98.80%	98.74%
17	talk.politics.guns	67.21%	65.90%	67.21%	63.93%	68.52%	66.55%
18	talk.politics.mideast	81.96%	76.27%	75.32%	78.80%	77.53%	77.98%
19	talk.politics.misc	47.88%	43.63%	43.24%	41.70%	43.24%	43.94%
20	talk.religion.misc	35.24%	38.57%	34.76%	33.81%	31.43%	34.76%
	Kemampuan rata - rata	**62.51%**	**63.51%**	**63.90%**	**62.62%**	**62.50%**	**63.01%**

Dari grafik 5.1 dan tabel 5.3 dapat disimpulkan bahwa kemampuan klasifikasi dokumen yang dimiliki oleh *SVM classifier* lebih baik dari pada *decision tree classifier*. Rata – rata kemampuan klasifikasi yang dimiliki SVM classifier adalah kurang lebih 94,20% sedangkan kemampuan decision tree classifier hanya 63,01 %.

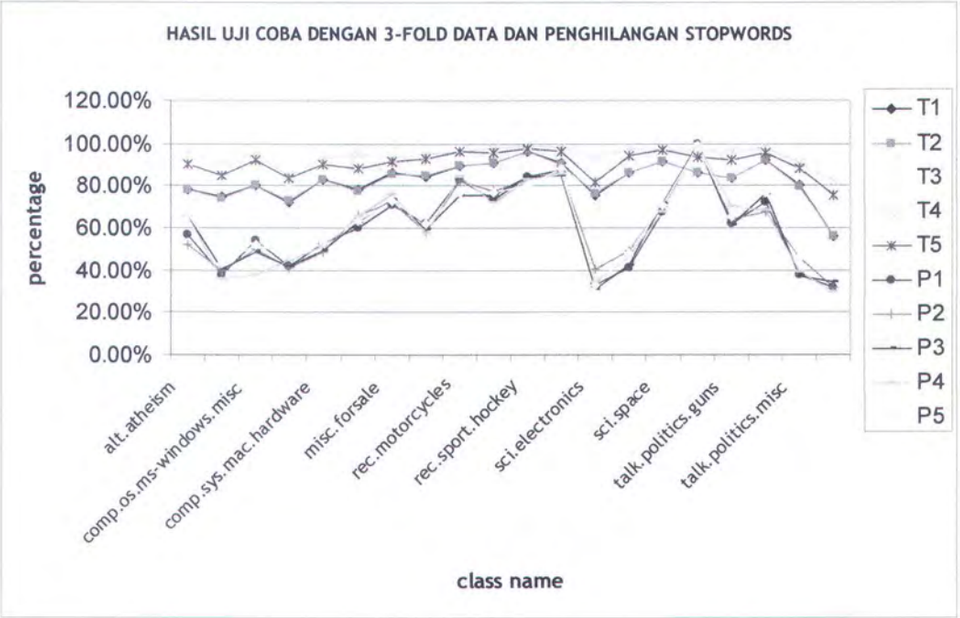
- **Uji coba II : dengan penghilangan *stopwords***

Uji coba berikut dilakukan dengan penghilangan *stopwords*, namun tanpa penggunaan stemming dan tanpa penggunaan thesaurus. Pada tabel 5.4 dan gambar 5.2 tetap adpat dilihat rata- rata kemampuan yang dimiliki oleh *SVM classifier* diatas rata – rata kemampuan *Decision Tree Classifier*. Dengan rata – rata kemampuan 88,40 %, sedangkan rata – rata pembanding hanya 61,25 %. Uji coba dilakukan sebanyak lima kali (H1 sampai dengan H5), serta dihitung pula kemampuan rata – rata dari lima kali uji coba tersebut. Hasil uji coba ini kemudian dibandingkan dengan data uji coba pada *decision tree classifier* (P1 sampai dengan P5), dapat dilihat pada tabel 5.4 dan gambar 5.2.

Tabel 5.4 Tabel perbandingan rata- rata kemampuan klasifikasi antara Dec. Tree classifier dengan SVM classifier, dengan 3-fold data validation, penghilangan stopword

Pesentase kemampuan rata – rata klasifikasi dokumen pada SVM Classifier							
	class name	**T1**	**T2**	**T3**	**T4**	**T5**	**rata - rata**
1	alt.atheism	77.90%	77.90%	91.76%	96.25%	90.64%	86.89%
2	comp.graphics	74.77%	74.46%	83.69%	90.46%	84.62%	81.60%
3	comp.os.ms-windows.misc	80.55%	80.55%	89.06%	95.14%	92.10%	87.48%
4	comp.sys.ibm.pc.hardware	71.95%	72.56%	80.49%	85.06%	83.54%	78.72%
5	comp.sys.mac.hardware	82.87%	82.87%	89.72%	93.15%	90.65%	87.85%
6	comp.windows.x	78.05%	77.74%	89.63%	95.12%	88.11%	85.73%
7	misc.forsale	86.42%	85.80%	90.74%	93.52%	91.36%	89.57%
8	rec.autos	84.55%	84.85%	92.42%	94.85%	93.03%	89.94%
9	rec.motorcycles	89.76%	89.46%	96.69%	96.08%	96.39%	93.67%
10	rec.sport.baseball	91.27%	91.27%	95.48%	97.59%	95.78%	94.28%
11	rec.sport.hockey	96.70%	96.70%	98.20%	99.10%	97.60%	97.66%
12	sci.crypt	90.94%	90.63%	96.68%	98.19%	96.37%	94.56%
13	sci.electronics	75.23%	76.15%	84.10%	93.27%	81.35%	82.02%
14	sci.med	86.06%	86.06%	91.52%	97.58%	94.55%	91.15%
15	sci.space	91.79%	91.79%	97.26%	97.26%	97.26%	95.08%
16	soc.religion.christian	86.49%	86.49%	96.10%	96.40%	93.99%	91.89%
17	talk.politics.guns	83.88%	83.55%	93.09%	96.38%	92.43%	89.87%
18	talk.politics.mideast	92.68%	92.68%	97.45%	98.73%	95.86%	95.48%
19	talk.politics.misc	79.92%	79.54%	88.80%	91.12%	88.42%	85.56%
20	talk.religion.misc	56.19%	55.71%	75.71%	82.38%	75.24%	69.05%
	rata - rata kemampuan	**82.90%**	**82.84%**	**90.93%**	**94.38%**	**90.96%**	**88.40%**
Pesentase kemampuan rata – rata klasifikasi dokumen pada Dec. Tree Classifier							
	class name	**P1**	**P2**	**P3**	**P4**	**P5**	**rata - rata**
1	alt.atheism	56.55%	51.69%	65.17%	65.54%	67.42%	61.27%
2	comp.graphics	37.61%	39.45%	40.67%	35.78%	42.81%	39.26%
3	comp.os.ms-windows.misc	53.78%	50.15%	48.34%	38.07%	51.66%	48.40%
4	comp.sys.ibm.pc.hardware	41.77%	40.85%	42.07%	44.51%	47.26%	43.29%
5	comp.sys.mac.hardware	52.57%	48.34%	49.24%	51.36%	53.47%	51.00%
6	comp.windows.x	59.94%	66.27%	62.65%	65.06%	63.25%	63.43%
7	misc.forsale	70.82%	72.34%	73.25%	75.99%	72.95%	73.07%
8	rec.autos	62.24%	58.01%	59.21%	57.10%	61.93%	59.70%
9	rec.motorcycles	83.18%	82.28%	75.68%	84.08%	76.58%	80.36%
10	rec.sport.baseball	73.65%	77.54%	75.45%	72.46%	76.65%	75.15%
11	rec.sport.hockey	84.13%	82.63%	84.43%	83.23%	79.94%	82.87%
12	sci.crypt	87.39%	86.79%	84.98%	85.29%	88.59%	86.61%
13	sci.electronics	32.83%	40.12%	31.00%	34.35%	32.83%	34.23%
14	sci.med	41.09%	49.55%	42.60%	46.53%	47.73%	45.50%
15	sci.space	67.58%	70.00%	71.21%	67.27%	70.61%	69.33%
16	soc.religion.christian	99.70%	99.40%	100.00%	99.70%	99.40%	99.64%
17	talk.politics.guns	62.30%	64.26%	61.31%	70.16%	65.25%	64.66%
18	talk.politics.mideast	72.15%	67.41%	76.27%	68.35%	76.58%	72.15%

19	talk.politics.misc	37.45%	45.56%	37.07%	39.00%	45.56%	40.93%
20	talk.religion.misc	31.90%	31.90%	34.29%	30.00%	42.38%	34.09%
	rata - rata kemampuan	**60.43%**	**61.23%**	**60.74%**	**60.69%**	**63.14%**	**61.25%**



Gambar 5.2 Hasil uji coba dan perbandingan klasifikasi data dengan 3-fold validation, penghilangan stopwords

- **Uji coba III : dengan penggunaan *stemming* saja**

Tabel 5.5 Tabel perbandingan rata- rata kemampuan klasifikasi antara Dec. Tree classifier dengan SVM classifier, dengan 3-fold data validation, dan penggunaan stemming

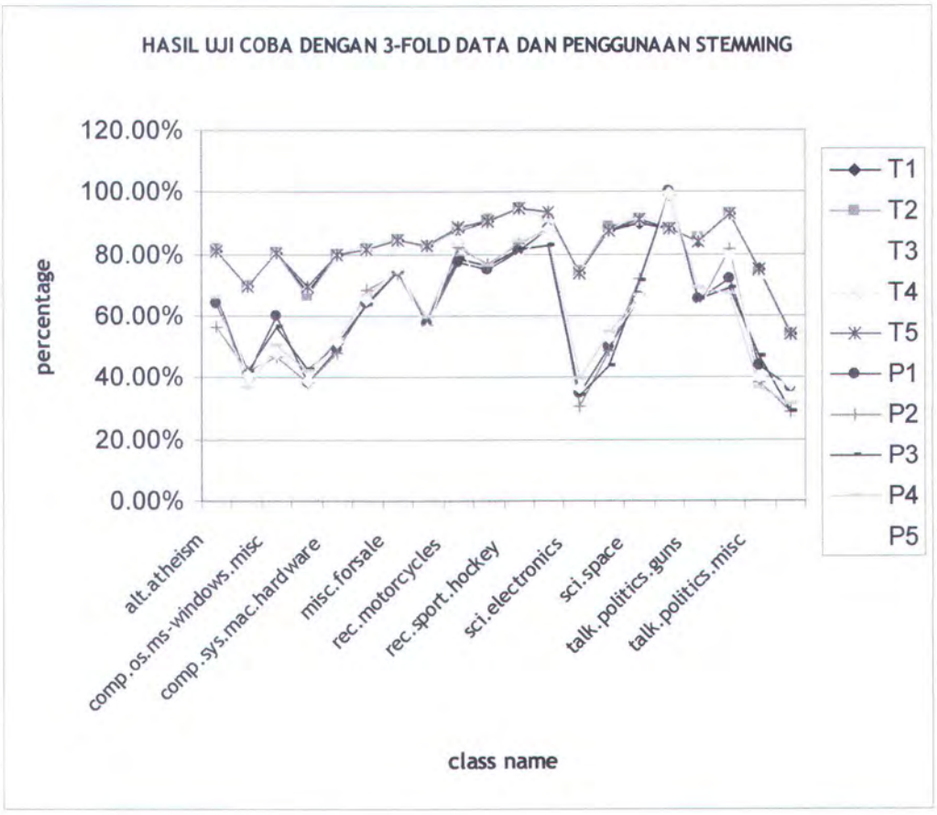
Pesentase kemampuan rata – rata klasifikasi dokumen pada SVM Classifier							
	class name	**T1**	**T2**	**T3**	**T4**	**T5**	**rata - rata**
1	alt.atheism	80.90%	81.57%	80.12%	82.02%	80.90%	81.10%
2	comp.graphics	69.68%	69.68%	69.54%	74.46%	69.54%	70.58%
3	comp.os.ms-windows.misc	80.24%	80.24%	80.24%	86.93%	80.24%	81.58%
4	comp.sys.ibm.pc.hardware	67.81%	66.83%	69.82%	69.51%	69.82%	68.76%
5	comp.sys.mac.hardware	79.75%	79.72%	79.75%	81.62%	79.75%	80.12%
6	comp.windows.x	81.71%	81.71%	81.71%	83.54%	81.71%	82.07%
7	misc.forsale	84.57%	84.57%	84.57%	82.72%	84.88%	84.26%

8	rec.autos	82.73%	82.43%	82.73%	81.21%	82.73%	82.36%
9	rec.motorcycles	87.53%	88.05%	88.55%	87.05%	88.55%	87.95%
10	rec.sport.baseball	90.36%	91.41%	90.36%	89.76%	90.36%	90.45%
11	rec.sport.hockey	94.89%	95.08%	94.89%	96.40%	94.89%	95.23%
12	sci.crypt	91.98%	90.97%	91.98%	92.15%	93.66%	92.15%
13	sci.electronics	74.01%	74.35%	74.01%	72.78%	74.01%	73.83%
14	sci.med	87.58%	88.53%	87.58%	87.88%	87.58%	87.83%
15	sci.space	90.12%	91.16%	91.19%	90.88%	91.19%	90.91%
16	soc.religion.christian	88.35%	88.35%	88.29%	87.39%	88.29%	88.13%
17	talk.politics.guns	84.21%	85.23%	85.23%	80.92%	84.21%	83.96%
18	talk.politics.mideast	92.99%	92.99%	92.99%	93.31%	92.99%	93.06%
19	talk.politics.misc	74.90%	73.90%	73.90%	70.66%	74.90%	73.65%
20	talk.religion.misc	53.81%	53.23%	53.23%	50.95%	53.81%	53.01%
	rata - rata kemampuan	**81.91%**	**82.00%**	**82.03%**	**82.34%**	**82.20%**	**82.10%**

Pesentase kemampuan rata – rata klasifikasi dokumen pada Dec. Tree Classifier

	class name	**PI**	**P2**	**P3**	**P4**	**P5**	**rata - rata**
1	alt.atheism	64.42%	56.18%	65.92%	65.92%	68.16%	64.12%
2	comp.graphics	40.98%	42.20%	41.90%	36.39%	40.98%	40.49%
3	comp.os.ms-windows.misc	60.12%	46.53%	56.19%	50.45%	47.73%	52.20%
4	comp.sys.ibm.pc.hardware	38.11%	38.11%	42.38%	42.07%	39.02%	39.94%
5	comp.sys.mac.hardware	48.94%	47.73%	51.06%	52.27%	51.36%	50.27%
6	comp.windows.x	64.16%	68.37%	63.86%	66.87%	65.66%	65.78%
7	misc.forsale	74.47%	73.56%	74.16%	74.77%	75.68%	74.53%
8	rec.autos	57.40%	57.10%	59.52%	59.52%	55.59%	57.83%
9	rec.motorcycles	77.18%	81.98%	78.38%	83.18%	84.68%	81.08%
10	rec.sport.baseball	74.85%	76.65%	76.05%	76.35%	63.77%	73.53%
11	rec.sport.hockey	81.14%	83.23%	81.74%	83.83%	78.74%	81.74%
12	sci.crypt	88.29%	88.29%	82.88%	87.69%	87.99%	87.03%
13	sci.electronics	34.65%	30.40%	34.35%	39.51%	36.47%	35.08%
14	sci.med	49.55%	48.34%	43.50%	54.38%	52.87%	49.73%
15	sci.space	66.06%	72.12%	71.21%	66.36%	65.15%	68.18%
16	soc.religion.christian	100.00%	99.10%	99.40%	99.40%	99.40%	99.46%
17	talk.politics.guns	65.57%	62.62%	65.57%	69.18%	62.30%	65.05%
18	talk.politics.mideast	71.84%	81.33%	69.30%	67.09%	79.43%	73.80%
19	talk.politics.misc	43.63%	38.22%	46.72%	36.68%	40.15%	41.08%
20	talk.religion.misc	36.67%	29.05%	28.57%	30.95%	37.62%	32.57%
	rata - rata kemampuan	**61.90%**	**61.06%**	**61.63%**	**62.14%**	**61.64%**	**61.67%**

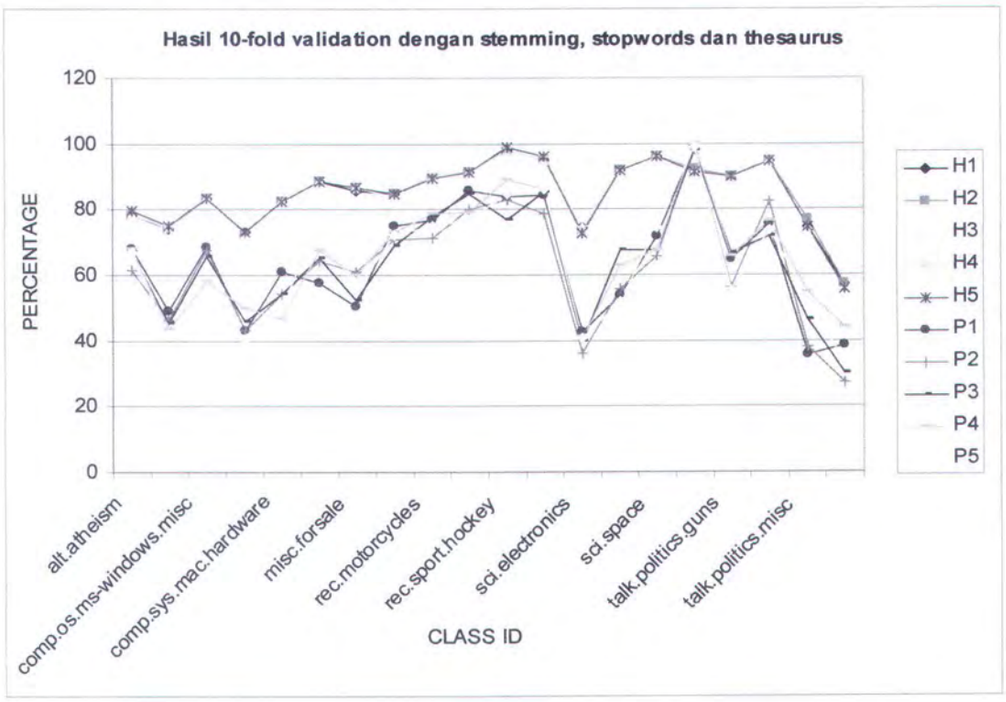
Pada uji coba kali ini, parameter indeks yang ditambahkan hanya penggunaan stemming saja, hasil uji coba dapat diamati pada tabel 5.5 dan gambar 5.3, dimana kemampuan klasifikasi *SVM classifier* masih diatas rata – rata kemampuan *Decision Tree Classifier*



Gambar 5.3 Tabel perbandingan rata- rata kemampuan klasifikasi antara Dec. Tree classifier dengan SVM classifier, dengan 3-fold data validation dan penggunaan stemming

Tampak pada kelas ke 17 atau '*talk.politics.guns*', kemampuan Dec. tree melebihi kemampuan SVM classifier, hal ini kemungkinan disebabkan karena untuk parameter trainning SVM_TRAIN masih digunakan setting parameter default. Sehingga rule – rule yang dihasilkan belum optimal mengenali semua kelas yang ada.

5.3.1.2 UJI COBA DENGAN 10-FOLD DATA VALIDATION



Gambar 5.4 Hasil uji coba dan perbandingan klasifikasi data dengan 10-fold validation, apply stemming, penghilangan stopwords dan penggunaan thesaurus

Uji coba berikut ini adalah uji coba dengan mengunakan data yang di validasi dengan 10-fold validation, tetapmenggunakan stemming, penghilangan stop words dan thesaurus. Perbedaan yang signifikan juga ditunjukkan dalam gambar 5.3, dan tabel 5.5, dimana SVM classifier memiliki kemampuan rata – rata 84,77 % sedangkan *decision tree classifier* hanya sekitar 63, 78 %. Uji coba dilakukan sebanyak lima kali (H1 sampai dengan H5), serta dihitung pula kemampuan rata – rata dari lima kali uji coba tersebut. Hasil uji coba ini kemudian dibandingkan dengan data uji coba pada *decision tree classifier* (P1 sampai dengan P5)

Tabel 5.6 Tabel perbandingan rata- rata kemampuan klasifikasi antara Dec. Tree classifier dengan SVM classifier, dengan10-fold data validation, penggunaan stemming, penghilangan stopword dan penggunaan thesaurus

Pesentase kemampuan rata – rata klasifikasi dokumen pada SVM Classifier							
	CLASS NAME	**H1**	**H2**	**H3**	**H4**	**H5**	**Rata - rata**
1	alt.atheism	78.41%	78.41%	78.41%	79.55%	79.55%	78.86%
2	comp.graphics	74.00%	74.00%	75.00%	74.00%	75.00%	74.40%
3	comp.os.ms-windows.misc	83.50%	83.50%	83.50%	83.50%	83.50%	83.50%
4	comp.sys.ibm.pc.hardware	73.00%	72.00%	72.00%	73.00%	73.00%	72.60%
5	comp.sys.mac.hardware	82.47%	82.47%	82.47%	82.47%	82.47%	82.47%
6	comp.windows.x	88.78%	88.78%	88.78%	88.78%	88.78%	88.78%
7	misc.forsale	85.86%	86.87%	86.87%	86.87%	86.87%	86.67%
8	rec.autos	84.85%	84.85%	83.84%	84.85%	84.85%	84.65%
9	rec.motorcycles	89.32%	89.32%	89.32%	90.29%	89.32%	89.51%
10	rec.sport.baseball	91.26%	91.26%	91.26%	91.26%	91.26%	91.26%
11	rec.sport.hockey	98.15%	98.15%	98.15%	98.15%	99.07%	98.33%
12	sci.crypt	95.00%	95.00%	95.00%	95.00%	96.00%	95.20%
13	sci.electronics	74.75%	73.74%	74.75%	73.74%	72.73%	73.94%
14	sci.med	91.92%	91.92%	91.92%	91.92%	91.92%	91.92%
15	sci.space	96.19%	96.19%	96.19%	96.19%	96.19%	96.19%
16	soc.religion.christian	91.51%	92.45%	91.51%	91.51%	91.51%	91.70%
17	talk.politics.guns	89.01%	90.11%	89.01%	90.11%	90.11%	89.67%
18	talk.politics.mideast	94.68%	94.68%	94.68%	94.68%	94.68%	94.68%
19	talk.politics.misc	74.39%	76.83%	74.39%	74.39%	74.39%	74.88%
20	talk.religion.misc	57.14%	57.14%	55.71%	55.71%	55.71%	56.29%
	Kemampuan rata - rata	**84.71%**	**84.88%**	**84.64%**	**84.80%**	**84.85%**	**84.77%**
Pesentase kemampuan rata – rata klasifikasi dokumen pada Des. Tree Classifier							
	CLASS NAME	**P1**	**P2**	**P3**	**P4**	**P5**	**Rata - rata**
1	alt.atheism	68.18%	61.36%	67.05%	68.18%	67.05%	66.36%
2	comp.graphics	49.06%	46.23%	45.28%	43.40%	41.51%	45.10%
3	comp.os.ms-windows.misc	68.22%	67.29%	65.42%	57.94%	53.27%	62.43%
4	comp.sys.ibm.pc.hardware	43.14%	43.14%	46.08%	50.00%	40.20%	44.51%
5	comp.sys.mac.hardware	60.75%	54.21%	54.21%	46.73%	50.47%	53.27%
6	comp.windows.x	57.43%	64.36%	65.35%	67.33%	76.24%	66.14%
7	misc.forsale	50.49%	61.17%	52.43%	60.19%	55.34%	55.92%
8	rec.autos	75.00%	71.00%	69.00%	73.00%	72.00%	72.00%
9	rec.motorcycles	76.85%	71.30%	77.78%	78.70%	75.00%	75.93%
10	rec.sport.baseball	86.00%	80.00%	85.00%	79.00%	77.00%	81.40%
11	rec.sport.hockey	84.00%	83.00%	77.00%	89.00%	85.00%	83.60%
12	sci.crypt	84.26%	78.70%	85.19%	86.11%	86.11%	84.07%
13	sci.electronics	42.86%	36.19%	40.00%	40.95%	40.95%	40.19%
14	sci.med	53.92%	55.88%	67.65%	62.75%	51.96%	58.43%
15	sci.space	71.72%	65.66%	67.68%	67.68%	74.75%	69.50%
16	soc.religion.christian	99.06%	99.06%	98.11%	100.00%	99.06%	99.06%
17	talk.politics.guns	64.58%	56.25%	66.67%	65.63%	55.21%	61.67%
18	talk.politics.mideast	75.49%	82.35%	71.57%	76.47%	73.53%	75.88%

19	talk.politics.misc	35.71%	38.10%	46.43%	54.76%	52.38%	45.48%
20	talk.religion.misc	38.57%	27.14%	30.00%	44.29%	32.86%	34.57%
	Kemampuan rata - rata	**64.26%**	**62.12%**	**63.90%**	**65.61%**	**62.99%**	**63.78%**

5.3.2 Uji Pengaruh Parameter Index

Kelompok uji coba berikut adalah untuk membandingkan perngaruh parameter indeks terhadap kamapuan rata – rata klasifikasi.

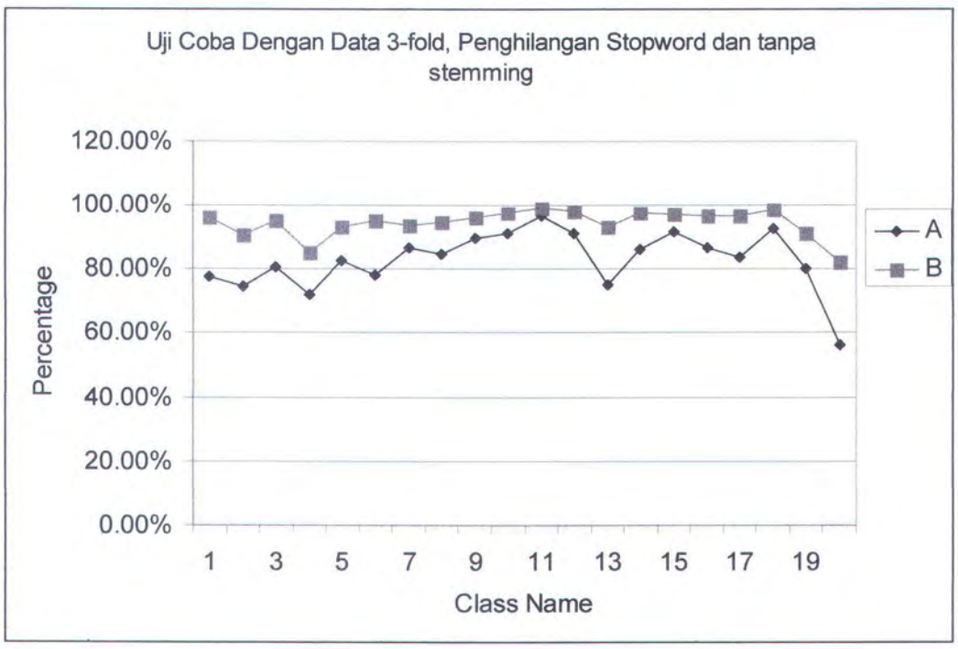
5.3.2.1 Uji Coba dengan 3-Fold Validation

Pada gambar 5.6 berikut ini terlihat adanya perbedaan yang cukup signifikan diantara dua grafik A dan B, grafik A adalah hasil uji coba klasifikasi dengan tanpa menghilangkan stop word, sedangkan grafik B adalah hasil uji coba dengan parameter indeks yang lengkap. Hal ini diakibatkan dengan tanpa dihilangkannya stopword kemampuan trainning menjadi semakin menurun karena rule yang dihasilkan semakin banyak. Bukti ini juga dapat dilihat pada sub bab sebelumnya yaitu dengan membadingkan hasil pada tabel 5.3 dan 5.4 (khusus untuk SVM classifier), dimana uji coba dengan parameter indeks lengkap memiliki rata – rata kemampuan klasifikasi 94 %, dan uji coba dengan tanpa penggunaan stemming dan thesaurus hanya sekitar 88,40 %.

Tabel 5.7 Perbandingan uji coba dengan penghilangan stopword dan tanpa penghilangan stop word

	Nama Kelas	A	B
1	alt.atheism	77.90%	96.25%
2	comp.graphics	74.77%	90.46%
3	comp.os.ms-windows.misc	80.55%	95.14%
4	comp.sys.ibm.pc.hardware	71.95%	85.06%
5	comp.sys.mac.hardware	82.87%	93.15%
6	comp.windows.x	78.05%	95.12%
7	misc.forsale	86.42%	93.52%
8	rec.autos	84.55%	94.85%
9	rec.motorcycles	89.76%	96.08%

10	rec.sport.baseball	91.27%	97.59%
11	rec.sport.hockey	96.70%	99.10%
12	sci.crypt	90.94%	98.19%
13	sci.electronics	75.23%	93.27%
14	sci.med	86.06%	97.58%
15	sci.space	91.79%	97.26%
16	soc.religion.christian	86.49%	96.40%
17	talk.politics.guns	83.88%	96.38%
18	talk.politics.mideast	92.68%	98.73%
19	talk.politics.misc	79.92%	91.12%
20	talk.religion.misc	56.19%	82.38%
	Rata-rata kemampuan	**82.90%**	**94.38%**



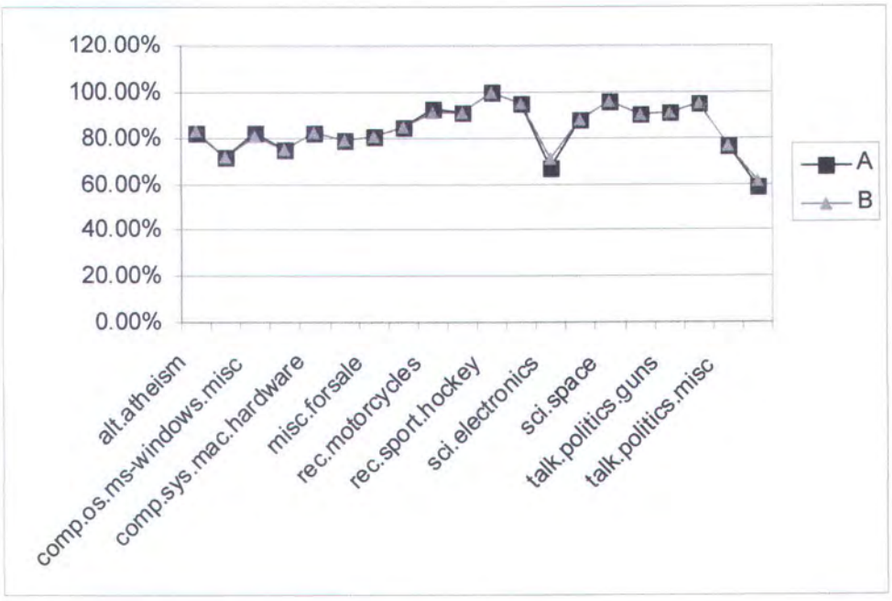
Gambar 5.5 Hasil uji coba Perbadingan penghilangan stop word dan tidak dihilangkan

5.3.2.2 Uji Coba dengan 10-Fold validation

Uji coba berikutnya pada 10-fold data, tampak tidak menunjukkan perbedaan yang berarti, antara penggunaan stemming dan tidak menggunakan stemming. Sata pada table 5.7 kolom A menunjukkan hasil uji coba dengan menggunakan stemming dan penghilangan stopword, sedangkan pada kolom B, hanya menggunakan penghilangan stopwords saja.

Tabel 5.8 Perbandingan uji coba dengan penggunan stemming dan tanpa penggunaan stemming

	Nama Kelas	A	B
1	alt.atheism	81.82%	82.95%
2	comp.graphics	72.00%	72.00%
3	comp.os.ms-windows.misc	82.52%	80.58%
4	comp.sys.ibm.pc.hardware	75.00%	75.00%
5	comp.sys.mac.hardware	82.47%	82.47%
6	comp.windows.x	78.57%	78.57%
7	misc.forsale	80.81%	80.81%
8	rec.autos	84.85%	84.85%
9	rec.motorcycles	92.23%	91.26%
10	rec.sport.baseball	91.26%	91.26%
11	rec.sport.hockey	100.00%	100.00%
12	sci.crypt	95.00%	95.00%
13	sci.electronics	66.67%	70.71%
14	sci.med	87.88%	87.88%
15	sci.space	96.19%	96.19%
16	soc.religion.christian	90.57%	90.57%
17	talk.politics.guns	91.21%	91.21%
18	talk.politics.mideast	94.68%	94.68%
19	talk.politics.misc	76.83%	76.83%
20	talk.religion.misc	58.57%	61.43%
	Rata-rata kemampuan	83.96%	84.21%



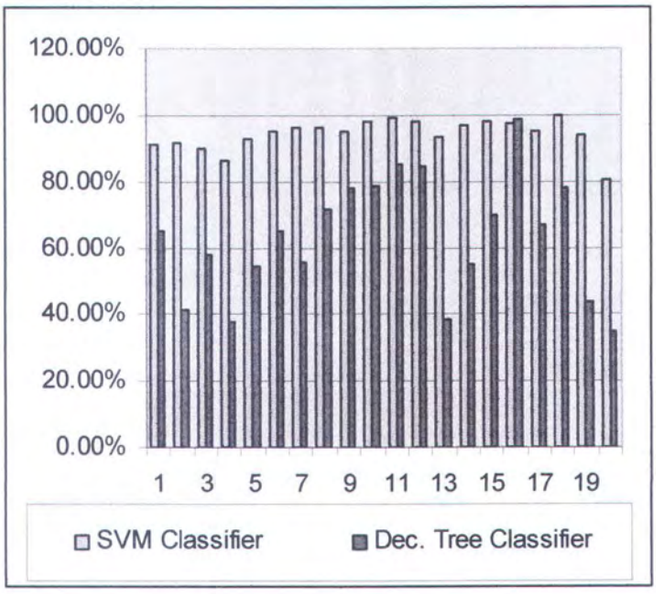
Gambar 5.6 Hasil uji coba Perbadingan penggunaan stemming dan tanpa stemming

5.3.3 EVALUASI UJI COBA

Dari hasil uji coba yang telah dilakukan, maka dapat dievaluasi dalam sub bab berikut ini.

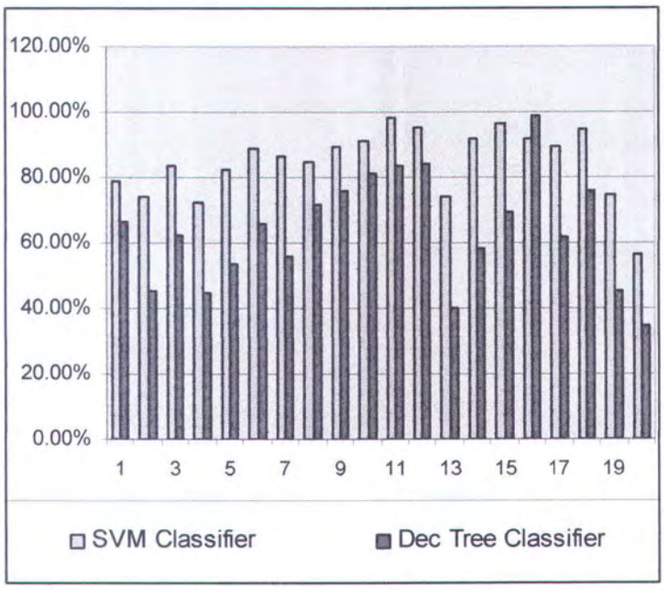
5.3.3.1 EVALUASI UJI COBA UNTUK MEMBANDINGKAN HASIL SVM CLASSIFIER DENGAN DEC. TREE CLASSIFIER

Pada uji coba ini dilakukan proses klasifikasi dokumen dengan SVM classifier dengan konfigurasi parameter indeks dibuat semirip mungkin dengan uji coba yang dilakukan oleh nuning, sedangkan untuk parameter trainning di-inisialisasi dengan nilai default-nya. Parameter indeks diset dengan penggunaan stemming, thesaurus dan penghilangan stopword. Untuk validasi 3-fold didapatkan hasil perbandingan pada gambar 5.7 (ditampilkan dalam bentuk diagram batang yang menunjukkan kemampuan rata – rata klasifikasi untuk setiap kelas, dimana terdapat 20 kelas) :



Gambar 5.7 Evaluasi perbandingan hasil uji coba pada dec. tree classifier dengan svm classifier untuk 3-fold data validation

Sedangkan untuk validasi data 10-fold juga dapat dilihat pada gambar 4 berikut :

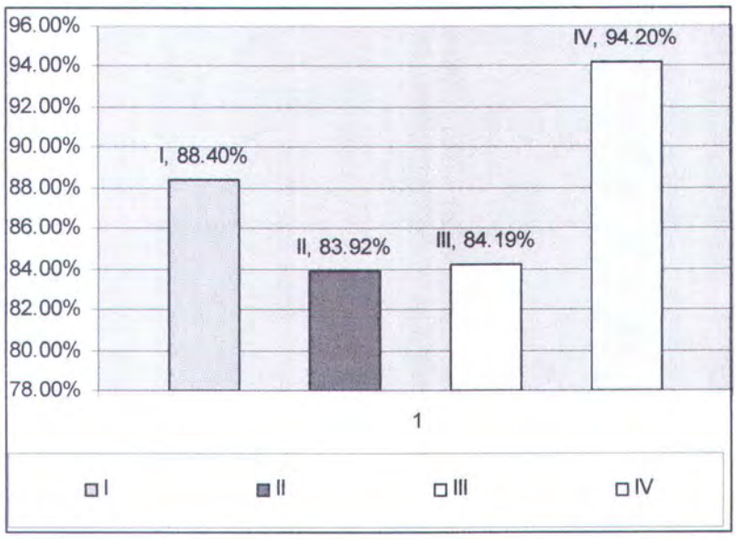


Gambar 5.8 Evaluasi hasil perbandingan uji coba pada dec. tree classifier dengan svm classifier untuk 10-fold data validation

Dari gambar 5.7 dan gambar 5.8 dapat disimpulkan, bahwa secara umum kemampuan klasifikasi yang dimiliki oleh *SVM classifier* masih lebih baik daripada *decision tree classifier*. Rata – rata kemampuan SVM classifier untuk dua uji coba diatas adalah diantara 84% - 94 %, sedangkan untuk Decision Tree Classifier hanya berkisar diantara 60 % - 64 %.

5.3.3.2 EVALUASI UJI COBA PENGARUH KONFIGURASI PARAMETER INDEKS TERHADAP KEMAMPUAN KLASIFIKASI

Berikutnya uji coba dilakukan untuk mengetahui pengaruh konfirgurasi parameter indeks terhadap kemampuan klasifikasi secara keseluruhan. Pada gambar 5.9 dapat diamati hasil uji coba bahwa penggunaan parameter indeks secara lengkap (stemming, penghilangan stopword dan thesaurus) akan dapat meningkatkan kemapuan klasifikasi.



Gambar 5.9 Evaluasi Hasil uji coba pengaruh konfigurasi parameter indeks

Keterangan :

I : Penggunaan Stemming saja

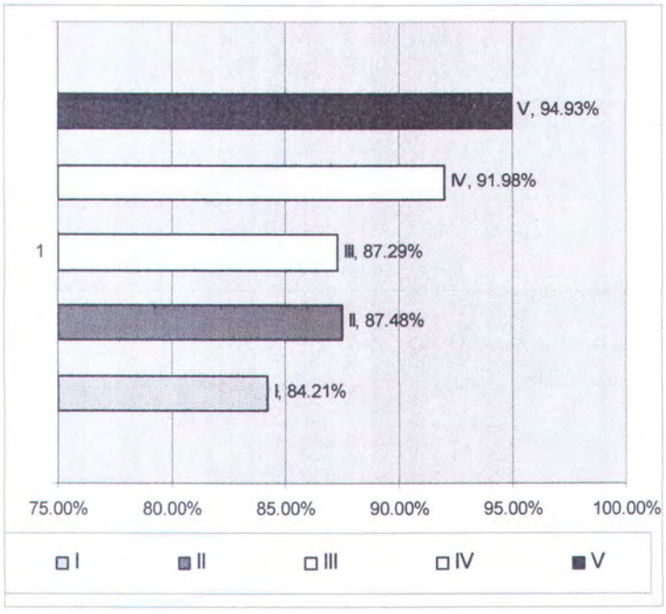
II : Penghilangan Stopword saja

III : Penggunaan Stemming dan Penghilangan Stopword

IV : Penggunaan Stemming, penghilangan Stopword dan penggunaan Thesaurus

5.3.3.3 EVALUASI UJICOBA PENGARUH KONFIGURASI PARAMETER TRAINING TERHADAP KEMAMPUAN KLASIFIKASI

Uji coba ketiga bertujuan untuk mengetahui pengaruh konfigurasi trainning terhadap kemampuan klasifikasi, parameter trainning yang dianalisa adalah MAX_FEATURES dan MAX_DOCTERMS. Hasil uji coba ditampilkan dalam bentuk diagram batang horisontal, seperti tampak pada gambar 5.9. Tampak dari hasil uji coba bahwa penentuan parameter trainning sangat berpengaruh terhadap kemampuan klasifikasi, penambahan jumlah MAX_DOCTERMS hanya berpengaruh sampai dengan nilai MAX_DOCTERMS = 500, hal ini mungkin disebabkan jumlah indeks kata / term pada dokumen hanya berkisar pada jumlah tersebut. Sedangkan pengaruh penambahan jumlah MAX_FEATURES akan sangat signifikan apabila nilai parameter ini diinisialisasikan pada nilai maksimumnya. Namun hal ini sangat berpengaruh pada jumlah waktu yang digunakan untuk testing (menjalankan operasi MACTHES), karena indeks kata pada sistem indeks berjumlah sangat besar.



Gambar 5.10 Evaluasi Hasil uji coba pengaruh konfigurasi parameter trainning

Keterangan :

I : Max. Features : 3.000; Max. Docterms : 50

II : Max. Features : 3.000; Max. Docterms : 500

III : Max. Features : 3.000 ; Max. Docterms : 5.000

IV : Max. Features : 10.000 ; Max. Docterms : 5.000

V : Max. Features : 100.000 ; Max. Docterms : 5.000

BAB VI

KESIMPULAN DAN SARAN

BAB VI
PENUTUP

Bab enam ini berisi kesimpulan yang dapat diambil setelah dilakukannya proses uji coba pada perangkat lunak. Dan kemudian, juga di tambahkan beberapa saran yang mungkin nantinya dapat digunakan untuk mengembangkan aplikasi ini lebih lanjut.

6.1 KESIMPULAN

Kesimpulan yang dapat diambil dari beberapa uji coba yang telah dilakukan, antara lain :

1. SVM Classifier terbukti memiliki kemampuan klasifikasi yang lebih baik daripada Decision Tree Classifier, dengan nilai selisih rata – rata kemampuan klasifikasi berkisar diantara 20 % sampai dengan 30 %.
2. Pada saat proses pembuatan indeks, parameter – parameter indeks sangat berpengaruh pada hasil testing.
3. Penentuan parameter trainning akan mempengaruhi kemampuan klasifikasi dan lamanya waktu yang dibutuhkan untuk melakukan trainning dan testing. Peningkatan nilai MAX_DOCTERMS dan MAX_FEATURES akan mengakibatkan naiknya kemampuan klasifikasi, namun juga menambah waktu yang digunakan dalam proses.

6.2 SARAN

Saran – saran yang dapat diberikan untuk pengembangan perangkat lunak ini antara lain :

1. Perlunya penanganan untuk dokumen dalam format yang lebih beragam sehingga kemampuan aplikasi ini dapat ditingkatkan.
2. Perlunya perbandingan dengan algoritma lain, sehingga pengukuran kemampuan klasifikasi dapat lebih beragam.
3. Perlunya uji coba untuk dilakukan klasifikasi pada dokumen multiformat

DAFTAR PUSTAKA

DAFTAR PUSTAKA

[Bae-1999] Baeza-Yates, R., Ribeiro-Neto, B. (1999). Modern Information Retreival. Addison Wesley

[Bur-1998] Burges, Christopher. (1998). *A Tutorial on Support Vector Machines for Patern Recognition. Data Mining and Knowledge Discovery* 2, 121 – 167.

[Fox-1992] Fox, C. (1992). Lexical Analysis and Stoplists *dalam* Frakes, W. B., Baeza-Yates, R. (ed.) *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, New Jersey.

[Fra-1992] Frakes, W. B. (1992). Stemming Algorithms dalam Frakes, W.B., Baeza-Yates, R. (ed.) *Information Retreival : Data Structures and Algorithms*. Prentice-Hall, New Jersey.

[Joa-1998] Joachims – Thorsten. *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*

[Lam-1996] Lam, L. Y., Savio, D. (1996). *Learned text Categorization by Backpropagation Neural Network*. Ph. M. Thesis, Departement of Computer Science, Hong Kong University of Science and Technology.

[Mit-1997] Mitchell, M. T. (1997) *Machine Learning*. The McGraw-Hill Companies, Inc.

[Ora-2003] Oracle Corporation. (2003). Oracle Text Reference 10g Release 1 (10.1)

[Orb-2003] Oracle Corporation. (2003). Oracle Text Application Developer's Guide, 10*g* Release 1 (10.1)

[Orc-2002] Oracle Corporation. (2002). Oracle Text 9.2.0 Technical Overview. Redwood Shores, CA

[Pur-2004] Purwaningsih Nuning,E.P. (2004) Perancangan dan Pembuatan Perangkat Lunak Pengklasifikasian Dokumen multiformat berbasis Oracle Text 9.2 dengan Algoritma Decision Tree Classifier, Jurusan Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember, Surabaya.

[Seb-2002] Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, Vol. 34(1), pp1-47

[Vap-1995] Vapnik.- Vladimir, N. The Nature of Statistical Learning Theory. Springer, New York, 1995.

[Zob-1989] Zobel, J., Moffat, A. (1998). Exploring the Similarity Space. Special Interest Group on Information Retrieval Forum, Vol. 32(1), pp18–34

LAMPIRAN

LAMPIRAN A

PREFERENCE BASIC LEXER PADA ORACLE TEXT 10.1

BASIC_LEXER digunakan untuk mengidentifikasikan token untuk membuat indeks teks untuk bahasa inggris dan bahasa lain yang memakai alfabet untuk bahasanya. BASIC_LEXER juga memungkinkan konversi ke huruf dasar, pengindeksan kata gabungan, pengindeksan huruf besar/ kecil, dan spelling alternatif untuk bahasa yang memiliki karakter set perpanjangan. Dalam bahasa Inggris dan Perancis, BASIC_LEXER juga dimungkinkan untuk pengindeksan tema.

Perhatikan juga bahwa setiap proses pada lexer yang dilakukan pada token sebelum indexing (misalnya, penghapusan karakter dan konversi ke huruf dasar juga dilakukan pada proses meng-query). Hal ini untuk memastikan query term akan sesuai dengan bentuk token pada indeks teks.

BASIC_LEXER mempunyai atribut sebagai berikut:

Atribut	Nilai Atribut	Keterangan
continuation	Karakter	Menspesifikasikan karakter yang mengindikasikan suatu kata yang bersambung pada baris berikutnya yang harus diindeks sebagai token tunggal. Karakter *continuation* yang umum adalah tanda hubung '-' dan backslash '\'.
numgroup	Karakter	Menspesifikasikan karakter tunggal yang seandainya muncul pada suatu string digit, mengindikasikan bahwa digit tersebut digrupkan pada unit tunggal yang lebih besar. Misalnya, koma ',' dapat didefinisikan sebagai karakter numgroup karena sering mengindikasikan penggrupan dari ribuan saat muncul distring digit.

numjoin	Karakter	Menspesifikasikan karakter yang saat muncul di string digits, mengakibatkan Oracle mengindeksnya sebagai unit tunggal kata. Sebagai contoh, titik '.' dapat didefinisikan sebagai karakter numjoin karena biasa muncul pada titik desimal saat muncul distring digit. Nilai default untuk numjoin dan numgroup ditentukan oleh inisialisasi Globalization Support yang dispesifikasikan pada database. Pada umumnya, nilai numjoin dan numgroup tidak perlu dispesifikasikan saat membuat lexer preference untuk BASIC_LEXER
printjoins	Karakter	Menspesifikasikan karakter non alfanumerik, yang saat muncul dimanapun dalam kata (diawal, tengah, atau akhir), diproses sebagai alfanumerik dan dimasukkan dengan token pada indeks teks. Ini juga termasuk printjoins yang sering muncul berturutan. Sebagai contoh, jika karakter tanda sambung '-' dan garis bawah '_' didefinisikan sebagai printjoins, kata-kata seperti *pseudo-intellectual* dan *_file_* akan disimpan pada indeks teks sebagai *pseudo-intellectual* and *_file_*. Jika karakter printjoins juga didefinisikan sebagai karakter punctuations, karakter hanya akan diproses sebagai karakter alfanumerik jika karakter mengikuti standar karakter alfanumerk atau telah didefinisikan sebagai karakter printjoins atau skipjoins
punctuations	Karakter	Menspesifikasikan karakter non-alfanumerik yang saat muncul diakhir kata, mengindikasikan akhir kalimat. Defaultsnya adalah titik '.', tanda tanya '?', dan tanda seru '!'. karakter yang didefinisikan sebagai punctuations akan dihilangkan dari token sebelum teks indexing. Bagaimanapun, jika suatu karakter punctuations juga didefinisikan sebagai karakter printjoins, karakter hanya dihilangkan jika karakter tersebut merupakan karakter terakhir dari token dan jika diikuti oleh karakter yang sama.

<table>
<tr><td></td><td></td><td>Sebagai contoh, misalnya titik (.) didefinisikan sebagai karakter printjoins dan punctuations, akan mengalami transformasi sebagai berikut saat proses indexing dan proses query:
<table>
<tr><th>Token</th><th>Token Terindeks</th></tr>
<tr><td>.doc</td><td>.doc</td></tr>
<tr><td>dog.doc</td><td>dog.doc</td></tr>
<tr><td>dog..doc</td><td>dog..doc</td></tr>
<tr><td>dog.</td><td>dog</td></tr>
<tr><td>dog...</td><td>dog..</td></tr>
</table>
Sebagai tambahan, BASIC_LEXER menggunakan karakter punctuations dalam penggabungan dengan karakter newline dan whitespace untuk menentukan deliminter kalimat dan paragraf untuk pencarian kalimat /paragraf</td></tr>
<tr><td>skipjoins</td><td>Karakter</td><td>Menentukan karakter non-alfanumerik, sehingga apabila karakter karakter itu muncul dalam suatu kata, dapat dibaca sebagai suatu token tunggal; bagaimanapun juga, karakter itu tidak disimpan dengan dengan token di indeks teks. Sebagai contoh, jika karakter tanda sambung '-' didefinisikan sebagai skipjoins, kata pseudo-intellectual disimpan pada indeks teks sebagai pseudointellectual. printjoins dan skipjoins merupakan mutually exclusive. Karakter yang sama tidak dapat dispesifikasikan dengan kedua atribut tersebut</td></tr>
<tr><td>Startjoins</td><td>Karakter non alphanumeric yang muncul pada awal</td><td>Startjoins menspesifikasikan karakter yang saat ditemukan sebagai karakter pertama dalam suatu token akan langsung diidentifikasikan sebagai awal</td></tr>
</table>

	token (string)	token. Karakter tersebut, sama halnya dengan karakter startjoins lain, akan langsung diikutkan dalam entri token indeks teks. Sebagai tambahan, karakter pertama startjoins dalam suatu string mengindikasikan akhir dari karakter pada string dari pada token sebelumnya
Endjoins	Karakter *non alphanumeric* yang muncul pada akhir *token* (string)	endjoins menspesifikasikan karakter yang saat ditemukan pada karakter akhir suatu token akan langsung mengidentifikasikan akhir token.karakter tersebut, seperti karakter *startjoins* lain, akan diikutkan pada entri token indeks teks. Aturan dibawah ini diaplikasikan untuk startjoins dan endjoins: • Karakter yang dispesifikasikan untuk startjoins/endjoins tidak boleh muncul pada atribut lain untuk BASIC_LEXER. • Karakter startjoins/endjoins hanya boleh muncul pada awal atau akhir token
whitespace	Karakter (string)	Menspesifikan karakter yang diperlakukan sebagai spasi kosong antar token. BASIC_LEXER menggunakan karakter whitespace dalam penggabungan dengan karakter punctuations dan newline untuk mengidentifikasikan karakter string yang berlaku sebagai pembatas pencarian kalimat atau paragraf. Nilai default yang didefinisikan untuk whitespace adalah 'space' dan 'tab'. Nilai tersebut tidak dapat diubah. Spesifikasikan karakter-karakter sebagai karakter whitespace untuk menambah default tersebut
newline	*NEWLINE (\n)* *CARRIAGE_RETURN (\r)*	Menspesifikasikan karakter yang mengindikasikan akhir dari baris pada teks. BASIC_LEXER menggunakan karakter newline dalam penggabungan dengan karakter punctuations dan whitespace untuk mengidentifikasikan karakter

		string yang diberlakukan sebagai pembatas untuk pencarian kalimat dan paragraf. Nilai valid satu-satunay untuk newline adalah NEWLINE dan CARRIAGE_RETURN. Defaultnya adalah NEWLINE
base_letter	NO (disabled) YES (enabled)	Menspesifikasikan karakter yang mempunyai karakter pengenal (umlauts, cedillas, acute accents, dan sebagainya) akan dikonvert ke bentuk dasar sebelum disimpan dalam indeks teks. Defaultnya adalah NO (konversi bentuk huruf dasar di-disabled).
mixed_case	NO (disabled) YES (enabled)	Menspesifikasikan jika lexer membiarkan token sebagaimana mereka muncul pada teks atau mengkonvert semua token ke huruf besar. Defaultnya adalah NO(semua token dikonvert ke huruf besar). Oracle menjamin bawah query kata akan sesuai dengan case sensitive dari indeks yang diquery. Sebagai hasilnya, jika meng-enable case sensitive untuk indeks teks, query pada indeks akan selalu case sensitive
composite	DEFAULT (tidak dilakukan pengindeksan kata gabungan, default) GERMAN (pengindeksan kata gabungan bahasa Jerman) DUTCH (pengindeksan kata gabungan bahasa Belanda)	Menspesifikasikan jika indexing kata gabungan akan di-disable atau di-enable untuk teks bahasa Jerman atau Belanda. Defaultnya adalah DEFAULT (indexing kata gabungan di-disabled). Kata-kata yang biasanya hanya dientrikan sekali pada kamus bahasa Jerman tidak dipisahkan ke *composite stems,* sedangkan kata-kata yang bukan entri kamus akan displit ke composite stems. Untuk mendapatkan composite stems yang terindeks, harus dibuat stem query, seperti *$bahnhof.* Bahasa wordlist stemmer harus sesuai dengan bahasa dari composite stems
index_stems	0 NONE	Menspesifikasikan stemmer untuk menggunakan stem indexing.Token kan distem ke bentuk dasar

	1 ENGLISH 2 DERIVATIONAL 3 DUTCH 4 FRENCH 5 GERMAN 6 ITALIAN 7 SPANISH	tunggal waktu diindeks, sebagai tambahan bentuk asalnya. Indexing stems memberikan performa query yang lebih baik untuk query stem ($) queries, seperti *$computed.*
index_themes	YES (enabled) NO (disabled, default)	Menspesifikasikan YES untuk mengindeks informasi tema dalam bahasa Inggris atau Perancis. Hal ini membuat query ABOUT akan lebih presisi. Atribut index_themes dan index_text attributes tidak bisa dibuat NO keduanya. Jika menggunakan BASIC_LEXER dan menspesifikasikan nilai NO untuk index_themes, maka default atribut ini akan NO. parameter ini bisa diset ke TRUE untuk semua tipe indeks termasuk CTXCAT. Untuk membuat query ABOUT dengan CATSEARCH, gunakan query template dengan grammar CONTEXT
index_text	YES (enabled, default NO (disabled)	Menspesifikasikan YES untuk mengindeks informasi kata. atribut index_themes dan index_text keduanya tidak bisa NO. defaultnya NO.
prove_themes	YES (enabled, default) NO (disabled)	Menspesifikasikan YES untuk pembuktian tema. Pembuktian tema berusaha mencari tema yang berhubungan dalam suatu dokumen. Saat tidak ada tema yang berhubungan ditemukan, tema parent akan dihilangkan dari dokumen. Jika pembuktian tema dapat diterima untuk dokumen besar, teks pendek berdeskripsi pendek dengan sedikit kata akan jarang menghasilkan pembuktian tema, dan mengakibatkan buruknya performa recall dengan query ABOUT. Pembuktian tema menghasilkan precision yang lebih

		tinggi dan recall yang lebih buruk(karena lebih sedikit baris yang dihasilkan) untuk query ABOUT. Untuk hasil recall yang lebih tinggi pada query ABOUT dan kemungkinan precision yang lebih buruk, pembuktian tema bisa didisable. Defaultnya YES. Atribut prove_themes hanya didukung untuk indeks CONTEXT dan CTXRULE
theme_language	AUTO (default) (Hanya bahasa yang didukung, bahasa Inggris dan Perancis)	Menspesifikasikan knowledge yang akan digunakan untuk pembuatan tema saat index_themes diset ke YES. Jika index_themes diset NO, men-setting parameter ini takkan menghasilkan apapun. Globalization Support language dapat didefinisikan sendiri ataupun AUTO. Knowledge base harus dimiliki untuk bahasa yang dispesifikasikan. Release ini hanya menyediakan knowledge base untuk bahasa Inggris dan Perancis. Untuk bahasa lain, bisa dibuat knowledge base sendiri. Defaultnya adalah AUTO, yang akan menginstruksi sistem untuk mengeset parameter ini sesuai bahasa di environment.
alternate_spelling	GERMAN DANISH SWEDISH NONE	Menspesifikasikan bahasa Jerman, Denmark, atau Swedia untuk memungkinkan alternate spelling dari salah satu bahasa tersebut. Memungkinkan alternate spelling akan memungkinkan proses query kata dari bentuk alternatifnya. Secara default, alternate spelling akan di enabled dalam semua tiga bahasa. NONE bisa dispesifikasikan untuk no alternate spelling

LAMPIRAN B

INSTALASI DATABASE ORACLE 10 G

Spesifikasi Perangkat Keras yang dibutuhkan :

1. Memori Fisik (RAM), minimal sebesar 256 MB, disarankan 512 MB
2. *Virtual Memory*, sebesar dua kali jumlah memori fisik
3. *Temp disk Space* sebesar 100 MB
4. Ruang pada Hard Disk 1,5 GB
5. Processor, minimal 200 MHz

Spesifikasi Perangkat Lunak yang dibutuhkan :

1. Sistem operasi, Database Oracle dapat berjalan pada sistem opeasi windows sebagai berikut :
   - Window NT Server 4.0, Windows NT Server Enterprise Edition, dan Terminal Server Edition dengan service pack 6a atau yang lebih tinggi, namun Oracle database tidak dapat dijalankan pada Windows NT Workstation.
   - Semua edisi Windows 2000 dengan service pack 1 atau yang lebih tinggi. Termasuk juga Terminal Services dan Windows 2000 MultiLanguage Edition (MLE).
   - Windows Server 2003
   - Windows XP Professional

2. *Compiler*, Pada Tabel A dapat dilihat berapa *library* yang disediakan Oracle untuk beberapa *system development* :

Table A. Library yang disediakan untuk beberapa Compiler

Compiler	System Development yang mendukung
Oracle C++ Call Interface	Microsoft C++ 6.0, Microsoft Visual C++ .NET 2002, dan Microsoft Visual C++ .NET 2003
Oracle Call Interface	Microsoft Visual C++ 6.0, Microsoft Visual C++ .NET 2002, dan Microsoft Visual C++ .NET 2003
External callouts	Microsoft Visual C++ 6.0, Microsoft Visual C++ .NET 2002, dan Microsoft Visual C++ .NET 2003
PL/SQL native compilation	Microsoft Visual C++ 6.0, Microsoft Visual C++ .NET 2002, dan Microsoft Visual C++ .NET 2003
Pro*COBOL	The Micro Focus Net Express compiler. Namun tidak mendukung untuk Object Oriented COBOL (OOCOBOL).
XDK	Microsoft Visual C++ 6.0, Microsoft Visual C++ .NET 2002, dan Microsoft Visual C++ .NET 2003

MILIK PERPUSTAKAAN
INSTITUT TEKNOLOGI
SEPULUH - NOPEMBER

3. Protokol jaringan, *Oracle Net foundation layer* menggunakan *Oracle protocol support* untuk berkomunikasi dengan protokol standar sebagai berikut : TCP/IP, TCP/IP dengan SSL, dan *Named Pipes*.

Cara melakukan instalasi perangkat lunak database oracle adalah sebagai berikut :

1. Masukkan CD-ROM dengan label Oracle Database 10g Release 1 (10.1) Disk 1 of 1 atau arahakan ke lokasi ke the Oracle Database 10g Release 1 (10.1) DVD-ROM. Untuk instalasi yang dijalankan dari hard-disk, dapat dilakukan dengan menjalankan file `setup.exe`. Dan untuk instalasi yang dijalankan dari media (CD-ROM / DVD-ROM), layar Autorun screen akan secara otomatis muncul jika tidak muncul jalankan langkah – langkah berikut :

   a. Klik Start > Run.

   b. Masukkan nilai : DRIVE_LETTER:\autorun\autorun.exe

   Setelah layar autorun muncul, klik tombol Install/Deinstall Products, sehingga muncul *Welcome screen*.

2. Tabel B dibawah ini menjelaskan langkah – langkah yang dapat diambil untuk setiap tampilan instalasi.

Table B. Petunjuk langkah instalasi untuk setiap tampilan instalasi

Tampilan Layar	Langkah yang disarankan
Welcome	Pilih *Basic Installation* atau *Advanced Installation*. Basic Installation digunakan untuk melakukan instalasi secara cepat. Metode instalasi ini hanya membutuhkan masukan user yang sedikit, dan tidak akan dijumpai layar dialog yang terlalu banyak Dengan metode ini akan dilakukan instalasi perangkat lunak dan pengguna dapat memilih untuk membuat basis data untuk tujuan umum, berdasarkan informasi yang telah ditentukan. Tekan tombol *next*.
Menentukan lokasi file	Pada bagian *Destination*, pengguna dapat menggunakan nilai awal yang sudah ditentukan atau dapat pula memasukkan nama Oracle Home dan lokasi direktori dimana akan di-instal komponen-komponen oracle. Tekan tombol *next*.
Pemilihan tipe instalasi	Pilih *Enterprise Edition*, *Standard Edition*, *Personal Edition*, atau *Custom*. Tekan tombol *next*.
Pemilihan konfigurasi basisdata	Pemilihan konfigurasi basisdata yang paling sesuai dengan kebutuhan pengguna. Tekan tombol *next*.

Penentuan pilihan konfigurasi basisdata	Informasi yang harus ditentukan antara lain : • Global Database Name Digunakan untuk menentukan nama dari basis data, dengan diikuti nama doimain dari sistem. • Database Character Set • Pembuatan basis data dengan skema contoh Tekan tombol *next*.
Pemilihan opsi manajemen basis data	Pengguna dapat menggunakan nilai awal yang sudah ada dan tekan tombol next.
Penentuan opsi file penyimpanan basisdata	Penentuan opsi sistem file dan specify the database file location, then Tekan tombol next. Penentuan lokasi basis data : Pengguna dapat menerima nilai awal yang sudah ditentukan atau memasukkan lokasi file yang baru.
Penentuan opsi Untuk *backup* dan *recovery*	Pengguna dapat menerima nilai awal yang telah ditentukan dan kemudian menekan tombol next.
Penentuan kata sandi	Penentuan kata sandi dan konfirmasi ulang untuk semua

untuk Skema basisdata	*account* basis data. Kemudian tekan tombol *next*
Catatan	Penelusuran kembali informasi instalasi yang telah dipilih, dan kemudian pengguna dapat menekan tombol *Install.*
Install	Penampilan status instalasi
Configuration Assistants	The Configuration Assistants screen displays status information for the configuration assistants that configure the software and create a database. After Database Configuration Assistant finishes, review the information on the screen. Make a note of the following information: • Enterprise Manager URL • Database creation logfiles location • Global Database Name • System Identifier (SID) • Server parameter filename and location Click OK to continue or click Password Management to unlock accounts and set passwords.
Akhir dari instalasi	Pada layar *configuration assistants* akan diatur beberapa palikasi berbasis Web, termasuk *Oracle Enterprise Manager Database Control.* Pada tampilan ini akan ditunjukkan alamat

	URL yang telah ditentukan untuk aplikasi tersebut. Juga dibuat catatan dari URL yang digunakan. Nomor port yang dipilih juga dicatat ke dalam file berikut : *ORACLE_BASE**ORACLE_HOME*\install\portlist.ini Untuk keluar dari Oracle Universal Installer, klik tombol *Exit*, dan kemudian tekan tombol *Yes*.

LAMPIRAN C

CONTOH DAN TAMPILAN PEMROSESAN DOKUMEN DENGAN ORACLE TEXT 10.1

Berikut ini akan dicontohkan beberapa langkah pemrosesan dokumen diatas platform Oracle Text 10.1. Juga akan ditunjukkan hasil serta tampilan dari setiap proses. Proses – proses yang akan dijalankan antara lain :

1. proses *loading* dokumen
2. pengindeks-an dengan *stemmer* dan tanpa *stemmer*
3. proses pengindeks-an untuk satu dokumen dan dua dokumen

Proses Loading dokumen diawali dengan pembuatan tabel penampung dokumen, pada contoh akan diberi nama DOC_TEST1, DOC_TEST2, dan seterusnya. Proses ini dijalankan dengan sintaks sql sebagai berikut :

```
1.  CREATE TABLE DOC_TEST1(
2.     DOC_ID NUMBER PRIMARY KEY,
3.     CLASS_ID NUMBER,
4.     FILE_NAME VARCHAR2(30),
5.     FORMAT VARCHAR2(30),
6.     TEXT CLOB)
```

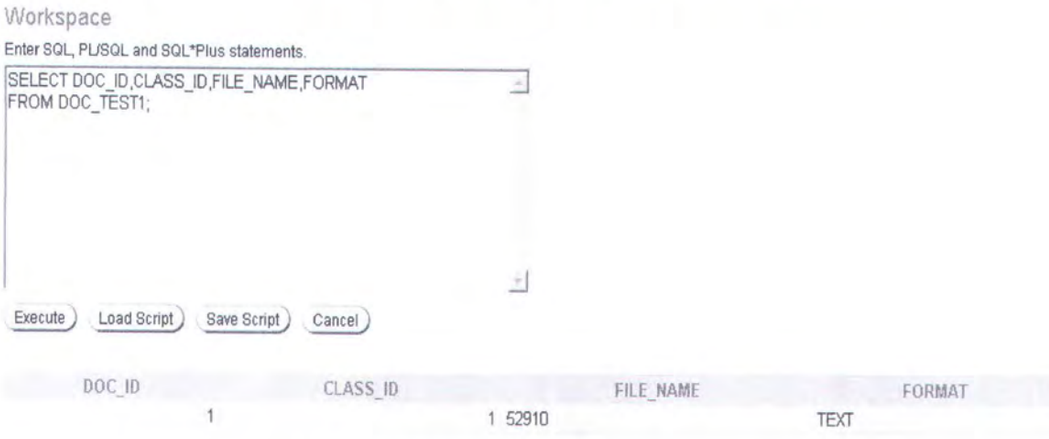
Langkah berikutnya adalah membuat direktori pointer yang akan mengarahkan Oracle untuk menunjuk langsung ke directory data yang akan diambil tersebut. Dengan perintah sebagai berikut :

```
CREATE OR REPLACE DIRECTORY MY_DIR AS 'D:\NUGIE\TA\DATA\20news-
19997\20_newsgroups\alt.atheism';
```

Menjalankan sintaks yang akan memasukkan file dokumen dengan nama 52910 ke dalam kolom text yang ada di tabel DOC_TEST1. Nilai kolom BLOB bisa diinisialisasi dengan menggunakan fungsi EMPTY_BLOB() sebagai predikat default. Sama saja, nilai suatu kolom CLOB atau NCLOB dapat diinisialisasikan dengan menggunakan fungsi EMPTY_CLOB().

```
1.  DECLARE
2.        F_LOB BFILE;
3.        C_LOB CLOB;
4.  BEGIN
5.  INSERT   INTO   DOC_TEST1(DOC_ID,CLASS_ID,FILE_NAME,FORMAT,TEXT)
6.  VALUES (1,1,'52910','TEXT',EMPTY_CLOB())
7.  RETURN TEXT INTO C_LOB;
8.        F_LOB := BFILENAME('MY_DIR','52910');
9.        DBMS_LOB.FILEOPEN(F_LOB,DBMS_LOB.FILE_READONLY);
10. DBMS_LOB.LOADFROMFILE(C_LOB,F_LOB,DBMS_LOB.GETLENGTH(F_LOB));
11.       DBMS_LOB.FILECLOSE(F_LOB);
12.       COMMIT;
13. END;
```

Hasil *loading* dokumen dapat diperiksa pada Oracle enterprise manager 10g berbasis web, dan akan didapatkan tampilan seperti pada gambar A berikut ini.



Gambar A. Hasil Loading Dokumen

Proses selanjutnya adalah, proses *indexing* dokumen, dengan maksud untuk menunjukkan hasil indeks dengan proses *stemming* dan tanpa proses *stemming*. Untuk melakukan proses *indexing* tanpa *stemmer index*, sebelumnya diatur terlebih dahulu index (CTX_DDL) preference beserta atributnya sebagai berikut :

```
1.  BEGIN
2.  ctx_ddl.drop_preference('MY_FILTER');
3.  ctx_ddl.create_preference('MY_FILTER', 'INSO_FILTER');
4.  ctx_ddl.drop_preference('MY_DATASTORE');
5.  ctx_ddl.create_preference('MY_DATASTORE', 'DIRECT_DATASTORE');
6.  CTX_DDL.DROP_STOPLIST('MY_STOPLIST');
7.  ctx_ddl.create_stoplist('MY_STOPLIST', 'BASIC_STOPLIST');
```

```
8.  ctx_ddl.add_stopclass('MY_STOPLIST', 'NUMBERS');
9.  CTX_DDL.DROP_SECTION_GROUP('MY_SECTION');
10. ctx_ddl.create_section_group('MY_SECTION', 'NULL_SECTION_GROUP');
11. ctx_ddl.drop_preference('MY_LEXER');
12. ctx_ddl.create_preference('MY_LEXER', 'BASIC_LEXER');
13. ctx_ddl.set_attribute('MY_LEXER', 'mixed_case', 'NO');
14. ctx_ddl.set_attribute ('MY_LEXER', 'index_text', 'YES');
15. ctx_ddl.set_attribute ('MY_LEXER', 'index_stems', 'NONE');
16. ctx_ddl.drop_preference('MY_WORDLIST');
17. ctx_ddl.create_preference('MY_WORDLIST', 'BASIC_WORDLIST');
18. ctx_ddl.set_attribute('MY_WORDLIST','STEMMER','NULL');
19. END;
```

Sedangkan untuk proses indexing dengan stem_index aktif sebelumnya diatur CTX_DDL preference beserta atributnya sebagai berikut :

```
1.  BEGIN
2.  ctx_ddl.drop_preference('MY_FILTER');
3.  ctx_ddl.create_preference('MY_FILTER', 'INSO_FILTER');
4.  ctx_ddl.drop_preference('MY_DATASTORE');
5.  ctx_ddl.create_preference('MY_DATASTORE', 'DIRECT_DATASTORE');
6.  CTX_DDL.DROP_STOPLIST('MY_STOPLIST');
7.  ctx_ddl.create_stoplist('MY_STOPLIST', 'BASIC_STOPLIST');
8.  ctx_ddl.add_stopclass('MY_STOPLIST', 'NUMBERS');
```

```
9.  CTX_DDL.DROP_SECTION_GROUP('MY_SECTION');
10. ctx_ddl.create_section_group('MY_SECTION', 'NULL_SECTION_GROUP');
11. ctx_ddl.drop_preference('MY_LEXER');
12. ctx_ddl.create_preference('MY_LEXER', 'BASIC_LEXER');
13. ctx_ddl.set_attribute('MY_LEXER', 'mixed_case', 'NO');
14. ctx_ddl.set_attribute ('MY_LEXER', 'index_text', 'YES');
15. ctx_ddl.set_attribute ('MY_LEXER', 'index_stems', 'ENGLISH
16. ctx_ddl.drop_preference('MY_WORDLIST');
17. ctx_ddl.create_preference('MY_WORDLIST', 'BASIC_WORDLIST');
18. ctx_ddl.set_attribute('MY_WORDLIST','STEMMER','ENGLISH');
19. END;
```

Proses *indexing* dijalankan dengan sintaks sebagai berikut :

```
1. CREATE INDEX DOC_TEST1_INDEXAB ON DOC_TEST1(TEXT) INDEXTYPE IS
2. CTXSYS.CONTEXT
3. PARAMETERS ('DATASTORE MY_DATASTORE LEXER MY_LEXER WORDLIST
4. MY_WORDLIST  STOPLIST MY_STOPLIST FILTER MY_FILTER');
5. COMMIT;
```

Dari kedua jenis aturan *preference* indeks diatas akan didapatkan hasilnya pada table `DR$DOC_TEST1_INDEXAB$I` dapat dilihat pada gambar B , dan `DR$DOC_TEST2_INDEXAB$I` pada gambar C.

Table Editor : "NUGIE"."DR$DOC_TEST1_INDEXAB$I" - nugie@ORCLASTE

TOKEN_TEXT	TOKEN_TYPE	TOKEN_FIRST	TOKEN_LAST	TOKEN_COUNT
BY	0	1	1	1
CAGE	0	1	1	1
CALSTATELA	0	1	1	1
CAN	0	1	1	1
CAPACITOR	0	1	1	1
CAPACITORS	0	1	1	1
CAPS	0	1	1	1
CHECK	0	1	1	1
CIRCUITRY	0	1	1	1
COMPARE	0	1	1	1
CRACKING	0	1	1	1
D	0	1	1	1
DEAD	0	1	1	1
DIDN	0	1	1	1
DIMENSIONS	0	1	1	1
DISTORTED	0	1	1	1
DOW	0	1	1	1
EDU	0	1	1	1

Execute time (s): 0.015 Rows returned: 152 Apply Revert Show SQL Close Help

Gambar B. Hasil Indeks dengan Index_stem tidak aktif pada tabel indeks $I

Table Editor : "NUGIE"."DR$DOC_TEST2_INDEXAB$I" - nugie@ORCLASTE

TOKEN_TEXT	TOKEN_TYPE	TOKEN_FIRST	TOKEN_LAST	TOKEN_COUNT
BUT	0	1	1	1
BY	0	1	1	1
CAGE	0	1	1	1
CALSTATELA	0	1	1	1
CAN	0	1	1	1
CAP	9	1	1	1
CAPACITOR	0	1	1	1
CAPACITOR	9	1	1	1
CAPACITORS	0	1	1	1
CAPS	0	1	1	1
CHECK	0	1	1	1
CIRCUITRY	0	1	1	1
COMPARE	0	1	1	1
CRACK	9	1	1	1
CRACKING	0	1	1	1
D	0	1	1	1
DEAD	0	1	1	1
DIDN	0	1	1	1

Execute time (s): 0.0 Rows returned: 180 Apply Revert Show SQL Close Help

Gambar C. Hasil Indeks dengan Index_stem aktif pada tabel indeks $I

Dari dua jenis hasil tersebut dapat diamati bahwa, apabila `index_stems` diaktifkan maka jumlah indeks akan bertambah (dapat dilihat pada informasi 'Rows returned'), jumlah indeks yang dihasilkan adalah 180, sedangkan apabila tidak diaktifkan hanya akan menghasilkan sebanyak 152 indeks kata. Sebagai contoh dapat dilihat pada kata 'CAPACITOR' dan kata 'CAPACITORS', pada teks dokumen sudah terdapat dua kata tersebut, ini dapat dilihat dari token_type kedua kata tersebut adalah sama dengan 0, namun pada tabel C muncul kata 'CAPACITOR' baru dengan

token_type tidak sama dengan 0. Token baru ini muncul akibat diaktifkannya index_stems, dan token baru ini merupakan kata dasar dari kata 'CAPACITORS'.

Uji coba berikutnya adalah dengan membandingkan hasil indeks dari satu dokumen dengan hasil indeks dari dua dokumen, langkah ini dijalankan dengan menambahkan dokumen baru pada tabel DOC_TES1. Hasil indeks yang didapatkan dapat dilihat pada gambar D. Proses *indexing* yang dijalankan adalah proses index tanpa `index_stem` aktif.

Table Editor : "NUGIE","DR$DOC_TEST2_INDEXAB$I" - nugie@ORCLASTE

TOKEN_TEXT	TOKEN_TYPE	TOKEN_FIRST	TOKEN_LAST	TOKEN_COUNT
C	0	2	2	1
C513WI	0	2	2	1
CAGE	0	1	1	1
CALSTATELA	0	1	1	1
CAN	0	1	1	1
CAPACITOR	0	1	1	1
CAPACITORS	0	1	2	2
CAPS	0	1	1	1
CHECK	0	1	1	1
CIRCUIT	0	2	2	1
CIRCUITRY	0	1	1	1
COLD	0	2	2	1
COMPARE	0	1	1	1
COMPLIANCE	0	2	2	1
COMPONENTS	0	2	2	1
CONTROLLED	0	2	2	1
COURSE	0	2	2	1
COVINGTON	0	2	2	1

Execute time (s): 0.016 Rows returned: 257 Apply Revert Show SQL Close Help

Gambar D. Hasil indeks dengan penambahan dokumen

Pada gambar D dapat diamati terdapat penambahan jumlah indeks yaitu menjadi 257 indeks (jika dibandingkan dengan gambar B, yang hanya terdapat 152 indeks). Hal ini diakibatkan adanya penambahan kata – kata baru dari dokumen baru yang dimasukkan.