PERANCANGAN DAN PEMBUATAN APLIKASI CLUSTERING TERM DOKUMEN MENGGUNAKAN ALGORITMA DIVISIVE INFORMATION-THEORETIC FEATURE CLUSTERING SEBAGAI METODE PENGURANGAN FEATURE PADA KLASIFIKASI DOKUMEN TEKS

TUGAS AKHIR

Disusun oleh :
Heru Eko Susanto
5199100092

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

PERANCANGAN DAN PEMBUATAN APLIKASI CLUSTERING TERM DOKUMEN MENGGUNAKAN ALGORITMA DIVISIVE INFORMATION-THEORETIC FEATURE CLUSTERING SEBAGAI METODE PENGURANGAN FEATURE PADA KLASIFIKASI DOKUMEN TEKS

TUGAS AKHIR

Diajukan untuk Memenuhi Sebagian Persyaratan
Memperoleh Gelar Sarjana Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui/Menyetujui

Dosen Pembimbing I,	Dosen Pembimbing II,
aac 4/8'05	
Rully Soelaiman, S.Kom. M.Kom.	Chastine Fatichah, S.Kom.
NIP. 132 303 065	NIP. 131 298 829

SURABAYA
Juli 2005

ABSTRAK

Salah satu penyebab tingginya waktu komputasi pada algoritma klasifikasi dokumen disebabkan oleh besarnya dimensi *feature* dari *training set*. Agar algoritma klasifikasi dapat bekerja dengan optimal maka perlu dilakukan pengurangan dimensi *feature*.

Metode pengurangan dimensi *feature* secara umum dapat dikelompokan menjadi dua kelompok utama, yaitu pengurangan dimensi *feature* dengan *feature selection* dan *feature clustering*. *Feature clustering* melakukan penggabungan *feature* menjadi beberapa klaster. Algoritma Divisive Information-Theoretic Feature Clustering adalah *distributional feature clustering* yang menggunakan distribusi statistik dari *feature*. Distribusi *feature* ini diperoleh dari dokumen yang digunakan sebagai *training set*. Algoritma ini menggabungkan *feature* ke klaster berdasarkan kedekatan setiap *feature* dengan klaster yang diukur dengan menggunakan Kullback-Leibler divergence. Algoritma ini juga merupakan algoritma klastering yang bertipe hard clustering. Klaster-klaster yang dihasilkan selanjutnya digunakan sebagai penganti *feature* dalam algoritma klasifikasi naïve bayes.

Uji coba dilakukan dengan menggunakan data UseNet Collection (20NG), yang terdiri dari 19997 dokumen. Dokumen-dokumen tersebut dikelompokan ke dalam 20 topic yang berbeda, dimana setiap dokumen hanya memiliki satu topic. Hasil uji coba klasifikasi dengan *word cluster* sebagai pengganti *feature* pada algoritma klasifikasi naïve bayes diperoleh hasil yang bervariasi. Dari hasil uji coba klasifikasi dapat diambil kesimpulan bahwa klasifikasi dengan *feature clustering* lebih baik daripada *feature selection* pada jumlah *feature* yang sedikit.

KATA PENGANTAR

Alhamdulillah, segala puji bagi Allah SWT yang senantiasa memberikan kekuatan hidayah dan limpahan kasih sayang-Nya. *Shalawat* dan salam tetap dicurahkan kepada *Rasulullah* Muhammad SAW. Akhirnya penulis bisa menyelesaikan tugas akhir ini.

Penulis memberi judul pada tugas akhir ini : **PERANCANGAN DAN PEMBUATAN APLIKASI CLUSTERING TERM DOKUMEN MENGGUNAKAN ALGORITMA DIVISIVE INFORMATION-THEORETIC FEATURE CLUSTERING SEBAGAI METODE PENGURANGAN FEATURE PADA KLASIFIKASI DOKUMEN TEKS.**

Semoga dengan selesainya tugas akhir ini, dapat membantu dalam proses pengembangan teknologi informasi di jurusan Teknik Informatika. Penulis menyadari bahwa tugas akhir ini jauh dari sempurna, masih banyak kekurangan-kekurangan. Oleh karena itu sangat diharapkan kritik dan saran yang membangun untuk perbaikan kedepan.

Akhirnya, penulis mengucapkan terima kasih kepada pihak-pihak yang telah membantu terselesaikannya tugas akhir ini.

Surabaya, Juli 2005

Heru Eko Susanto

UCAPAN TERIMA KASIH

Alhamdulillah, segala puji hanya untuk Allah SWT yang selalu melimpahkan rahmat dan hidayah serta banyak kemudahan didalam mengerjakan Tugas Akhir ini. *Shalawat* serta *salam* tetap tercurahkan kepada Rasulullah Muhamad SAW. Akhirnya penulis bisa menyelesaikan tugas akhir ini.

Saya ingin mengucapkan terima kasih dan penghargaan sebesar-besarnya kepada semua pihak yang telah membantu pelaksanaan Tugas Akhir ini baik secara langsung maupun tidak langsung kepada:

1. Orang tua penulis, serta adik-adik penulis dwi dan huda yang telah memberikan perhatian, nasehat, bantuan material dan spritual, serta doa yang tidak ternilai besarnya demi kelancaran tugas akhir ini. Demikian juga untuk keluarga yang ada di Tulungagung dan Surabaya, Mbok Tun, Mbok Sum, Kang Yana sekeluarga, Mak Ni & Pak Lis sekeluarga, Ida & Mas Narmo sekeluarga.
2. Bapak Rully Sulaiman S.Kom, M.Kom dan Ibu Chastine Fatichah S.Kom selaku pembimbing Tugas Akhir yang senantiasa sabar dalam memberikan petunjuk dan bimbingan.
3. Bapak Ir. Yudhi Purwananto M.Kom selaku Ketua Jurusan Teknik Informatika ITS sekaligus Dosen Wali yang telah membimbing selama kuliah.
4. Bapak Prof. Dr. Ir. Arif Djunaidy M.Sc selaku Dekan Fakultas Teknologi Informasi ITS.
5. Bapak dan Ibu Dosen yang telah membagikan ilmunya selama saya kuliah di Teknik Informatika ITS.

6. Seluruh karyawan dan staf Fakultas Teknologi Informasi ITS; Pak Yudi, Pak Narno, Pak Mu'in, Mba Eva, Mbak Davi (sekarang di Perpus Pusat) serta karyawan yang lain atas bantuannya selama penulis kuliah.
7. Budi Nugroho sekeluarga (suwun bantuan printernya cak), Ary "Pak Dosen"(thanks untuk sharingnya), Kholid (sabar ya mas), Wahib (badminton yuk), Arieq (terimakasih banyak membantu), Sujianik (trim's untuk obrolan sore harinya), Mimiek, Ade Reza, Ridha, Nailul, Imam serta seluruh keluarga besar angkatan 99/C0F, terimakasih atas kebersamaan dan hari-hari indah yang telah kita lewati.
8. Rekan-rekan di Global Performa (Pak Patdono, Pak Edi, Mas Andik, Yudhi, Mas Anton dan Mbak Fita), rekan-rekan di Otomasi (Mbak Resty, Mas Ucok, Mas Yanto, Mas Yono, Dul, Alan, Mas Bagus) yang telah berbagi ilmu dan pengalaman selama ini.
9. Mbak Ummy Nafisah, Sri Giyanti (trim's untuk smes2nya dan traktiran gudeg) terima kasih atas dorongan semangat, obrolan seru serta jogjanya.
10. Eko Yusuf (suwun atas persahabatannya selama ini), Dwi Sukendar, Eka Nuryanti (+ Ita), Dwi Susi Andriani, Septia Eka, Wahyuning terimakasih atas perjumpaannya kembali.
11. Teman-teman kost di keputih IA-22, geng Tulungagung (mbah heri, iwan, tomi), hantoko (trim's bantu ngangkat komputer ke kampus).
12. Teman-teman kelas 2-1 SMU Boyolangu Tulungagung Iwan, Okta, Ridha, Lutfhi, Lia, serta semua pihak yang tidak mungkin saya sebutkan satu-persatu yang telah banyak membantu selama ini

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Pada bab ini akan dijelaskan tentang hal-hal yang melatarbelakangi pembuatan tugas akhir, tujuan, permasalahan yang dihadapi serta batasan masalah dan metodologi pembuatannya

1.1. LATAR BELAKANG

Perkembangan dokumen teks, yang meningkat secara eksponensial baik dalam jumlah dokumen maupun orang yang membuat maupun memakai dokumen, dipengaruhi oleh perkembangan internet.. Informasi tentang perusahaan tidak lagi berupa angka-angka, melainkan juga disimpan dalam bentuk teks dalam dokumen, halaman web, buku-buku manual, laporan-laporan keuangan, *email*, faksimili dan presentasi. Sumber-sumber informasi strategis ini sering kali tidak dapat dimanfaatkan dengan baik karena tidak adanya pengelolaan yang baik terhadap dokumen-dokumen tersebut.

Pengelolaan dokumen terstruktur akan membantu menganalisa, navigasi, pencarian (*searching*) dan perawatan dokumen. Dokumen-dokumen teks tersebut dikelompokkan atau diklasifikasikan berdasarkan tema atau topik yang sama. Pengklasifikasian dokumen (*Document Classification*) adalah metode yang dilakukan secara manual atau otomatis untuk menempatkan dokumen dalam kategori berdasarkan tema (*class label*) yang ada pada dokumen tersebut [Pratt – 1999]. Keberadaan perangkat lunak pengkategorian dokumen secara otomatis

menjadi *tool* yang penting didalam membantu mengelola dokumen-dokumen tersebut.

Berbagai macam algoritma pengklasifikasian dokumen telah dikembangkan oleh para ahli. Algoritma pengklasifikasian dokumen dapat dibagi menjadi 2 macam menurut metode klasifikasinya yaitu: *supervised learning algorithm* dan *unsupervised learning algorithm*. Algoritma yang termasuk dalam *supervised learning algorithm* adalah *decision-rule induction, decision-tree induction, nearest-neighbor algorithm, bayesian classifier, discriminant analysis* dan *neural network*.

Secara umum pada sistem klasifikasi yang dengan metode *supervised learning algorithm* digunakan langkah-langkah dasar yang sama untuk memproses klasifikasi dokumen. Pada *Supervised Learning Algorithm* digunakan sekumpulan dokumen yang sudah terlabeli dalam beberapa *class label* sebagai *training set*. Setiap dokumen dalam *training set* direpresentasikan sebagai sebuah kumpulan *term-term* yang membentuk vektor (*vector of terms*). Setiap elemen *i* dalam vektor adalah representasi nilai dari *term i*. Setiap vektor dokumen adalah sebuah koordinat titik dalam sebuah ruang *term* multidimensi (*multidimensional term space*). Paradigma ini memberikan cara pandang yang intuitif didalam melihat dokumen sebagai sebuah titik koordinat dalam ruang multidimensi, dimana kemiripan diantara 2 dokumen ditentukan berdasarkan kedekatan jarak antara keduanya.

Dengan menggunakan *term* sebagai representasi sebuah dokumen, maka dokumen-dokumen dalam *training set* sering kali berupa *high dimensional feature*

space yang tersebar [Dhill–2003]. Tingginya dimensi dari *feature space* berakibat pada tidak optimalnya suatu algoritma klasifikasi dokumen. Agar algoritma klasifikasi dapat bekerja secara optimal maka pengurangan term perlu dilakukan. Prosedur standar yang dilakukan untuk mengurangi tingginya dimensi dari *feature space* adalah dengan pemilihan *term* (*term selection* atau yang sering dikenal dengan *feature selection*). *Term selection* memilih sebagian dari *term* dengan memakai kriteria permilihan *term* yang telah didefinisikan sebelumnya. *Term-term* yang dipilih tersebut selanjutnya digunakan sebagai *feature* pada algoritma klasifikasi. Prosedur lain yang bisa digunakan untuk mengurangi tingginya dimensi *feature* dalam *information retrieval* adalah dengan *latent semantic indexing* dan *probabilistic LSI*.

Alternatif lain yang digunakan untuk mengurangi tingginya dimensi *feature space* adalah dengan mengelompokkan *term-term* yang dianggap "sama" dalam sejumlah klaster kata (*word cluster*). Dari klaster yang dihasilkan tersebut selanjutnya digunakan sebagai pengganti *term/feature* pada algoritma klasifikasi. Metode klastering kata telah banyak digunakan dalam pemodelan bahasa (*language modeling*) dan *word co-occurrence* [Baker-1998]. Klaster kata terdistribusi (*distributional word cluster*) mengelompokkan kata-kata yang memiliki kesamaan fungsi distribusi probabilitas [Baker-1998]. Riset yang dilakukan oleh Douglas L Baker & Andrew Mc Callum [Baker-1998] diperoleh kesimpulan bahwa dengan metode *distributional clustering* dapat dihasilkan performa klasifikasi yang baik dalam pengklasifikasian dokumen. Performa klasifikasi dapat ditingkatkan dengan *clustering* ketika distribusi *training data*

tersebar, dimana statistik rata-rata untuk kata-kata yang sama dihasilkan estimasi yang lebih akurat [Baker–1998]. Untuk mengukur kemiripan antara disribusi probabilitas bisa digunakan berbagai varian dari *kullback-leibler divergence*. Masih menurut Douglas Baker [Baker-1998] *word clustering* memiliki beberapa kelebihan yaitu (1) dihasilkan *useful semantic word clustering*, (2) meningkatkan keakurasian klasifikasi, (3) jumlah model klasifikasi yang lebih kecil.

1.2. PERMASALAHAN

Adapun permasalahan yang akan dihadapi dalam pembuatan tugas akhir ini adalah :

1. Bagaimana *preprocessing training set* dilakukan.
2. Bagaimana membangun aplikasi dengan platform java dengan database MySQL.
3. Bagaimana algoritma Divisive Information Theoretic Feature Clustering dimplementasikan untuk memperoleh *word cluster* yang digunakan sebagai *feature*.
4. Bagaimana memakai *word cluster* sebagai pengganti *term* pada algoritma klasifikasi *Naïve Bayes*.

1.3. BATASAN MASALAH

Tugas akhir ini ditekankan pada implementasi perangkat lunak pengklasteran kata/*term* dengan batasan masalah.

1. Digunakannya platform java dan database MySQL serta Hibernate sebagai *object relational persistence mapping*.

2. Dokumen yang digunakan adalah dokumen teks berbahasa inggris.
3. Pengklasifikasian dokumen yang digunakan untuk menguji *distributional word cluster* dengan menggunakan metode *naïve bayes classifier*.

1.4. TUJUAN

Tujuan dari pembuatan tugas akhir ini adalah membangun aplikasi pengklasifikasian dokumen dengan memakai algoritma Divisive Information Theoretic Feature Clustering untuk memperoleh *word cluster* sebagai pengganti *feature* dalam klasifikasi dokumen.

1.5. METODOLOGI PEMBUATAN TUGAS AKHIR

Pembuatan tugas akhir ini dilakukan dengan metodologi sebagai berikut:

1.5.1. Studi Literatur

Pada tahapan ini dipelajari paper-paper seputar motode-metode pengurangan dimensionalitas feature dalam klasifikasi dokumen, pemrograman java berorientasi obyek, hibernate dan algoritma klasifikasi dokumen dengan metode *naïve bayes*.

1.5.2. Perancangan Sistem dan Aplikasi

Pada tahap ini dilakukan perancangan perangkat lunak berdasarkan literature yang telah dipelajari meliputi penentuan struktur data yang digunakan, proses-proses yang akan dilakukan. Perancangan sistem ini mengacu pada langkah-langkah yang digunakan untuk mendesain sistem berorientasi obyek dengan metode *unified modeling language* (UML) memakai tool Rational Rose

- Pengumpulan data

  Pada tahapan ini dilakukan pencarian dan pemilihan kata dokumen berbahasa inggris untuk digunakan sebagai *dataset* yang dipilih data yang berasal dari 20 *Newsgroups (20 NG)*, yang telah dikumpulkan oleh Ken Lang. *Dataset* ini berisi hampir 20 ribu artikel yang dibagi menjadi dalam 20 grup diskusi Usenet. Setiap grup adalah satu kategori (*class label*) dalam klasifikasi.

- Perancangan proses

  Pada tahap ini akan ditentukan hal-hal yang akan dilakukan dan dibutuhkan oleh sistem :

  - Database MySQL sebagai penyimpan data dari aplikasi yang dibangun.
  - Hibernate sebagai *object relational persistence mapping* aplikasi dengan data dari database.
  - Pembuatan aplikasi *feature clustering* dengan algoritma Divisive Information Theoretic Feature Clustering. Input yang digunakan oleh perangkat lunak ini adalah dokumen teks yang diperoleh pada tahap pengumpulan data. Sedangkan output yang dihasilkan adalah klaster-klaster kata *term*. Untuk menguji hasil klaster digunakan algoritma klasifikasi *naïve bayes*.

- Perancangan antar muka

  Pada tahap ini dibuat rancangan antar muka dari aplikasi *feature clustering* dan *document classification*.

1.5.3. Pembuatan Aplikasi

Pada tahap ini dilakukan implementasi pembuatan aplikasi berdasarkan rancangan yang telah dibuat pada tahap sebelumnya.

1.5.4. Uji Coba dan Evaluasi

Pada tahap ini dilakukan pengujian terhadap perangkat lunak dengan dokumen yang telah diperoleh pada tahap sebelumnya untuk memperoleh klaster kata. Algoritma klasfikasi *naïve bayes* digunakan untuk menguji klasifikasi dokumen dengan klaster yang telah dihasilkan.

1.5.5. Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan dan dokumentasi perangkat lunak dari tahap pendahuluan hingga tahap kesimpulan dan saran.

1.6. SISTEMATIKA PEMBAHASAN

Sistematika pembahasan yang digunakan dalam tugas akhir ini adalah sebagai berikut:

1. Bab I, Pendahuluan. Bab ini berisi latar belakang, permasalahan, tujuan. Batasan masalah, metodologi, dan sistematika pembahasan.
2. Bab II, Teori Informasi. Bab ini berisi tentang konsep Teori informasi menjadi dasar framework algoritma Divisive Information Theoretic Feature Clustering.
3. Bab III, Penggunaan Word Cluster Pada Klasifikasi Dokumen, Bab ini membahas mengenai *distibutional word clustering* serta peranannya dalam klasifikasi dokumen. Selain itu juga dibahas teori

pengklasifikasian dokumen, termasuk didalamnya klasifikasi dengan *naïve bayes*.

4. Bab IV, Perancangan Perangkat Lunak. Bab ini membahas tentang perancangan perangkat lunak yang digunakan untuk memperoleh *word cluster* serta klasifikasi dokumen.
5. Bab V, Implementasi Perangkat Lunak. Pada bab ini berisi pembahasan mengenai implementasi perangkat lunak.
6. Bab VI, Uji Coba dan Evaluasi Sistem. Pada bab ini dilakukan serangkaian uji coba dan pembahasan tentang hasil ujicoba
7. Bab VII, Kesimpulan dan saran. Pada bab ini berisikan kesimpulan yang diperoleh dari pengujian perangkat lunak serta saran-saran yang digunakan untuk perbaikan

BAB II
TEORI INFORMASI

Bab ini membahas teori informasi yang menjadi konsep dasar yang digunakan dalam membentuk *distributional word clustering* dalam algoritma *divisive feature clustering*

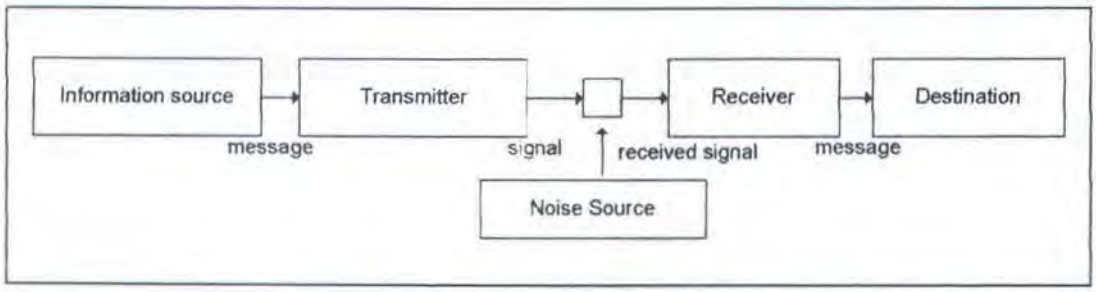
2.1. PENDAHULUAN

Teori Informasi pertama kali dipublikasikan pada jurnal *the Bell System Technical Journal vol. 27, pp 379-423, 623-656, July, October,* 1948. yang berjudul *A Mathematical Theory of Communication* oleh seorang peneliti yang bernama CE Shannon yang saat itu berkerja pada Bell Laboratorium. Paper tersebut membahas bagaimana memodelkan komunikasi kedalam notasi matematika. Teori informasi juga membahas bagaimana melakukan kompresi *message* agar dapat ditransmisikan secara efisien. Dalam Tugas Akhir ini hanya akan dibahas penggunaan Teori Informasi pada klasifikasi dokumen

2.2. SISTEM KOMUNIKASI

Manusia berkomunikasi dengan memakai simbol-simbol. Simbol-simbol tersebut bisa berupa gambar atau suara. Kemampuan komunikasi manusia dimulai sejak jaman prasejarah dimana manusia mencoba untuk berkomunikasi dengan menulis apa-apa yang dilihatnya pada dinding-dinding gua. Sekarang simbol-simbol komunikasi suara dan gambar ditransmisikan dalam bentuk kode-kode sinyal-sinyal digital keberbagai belahan dunia dengan kecepatan yang hampir

mendekati kecepatan cahaya. Bahkan para periset di NASA dan lembaga penelitian luar angkasa lainnya mencoba untuk mengirim sinyal-sinyal komunikasi ke luar angkasa dengan harapan dapat ditemukan bentuk kehidupan lain di alam raya.



Gambar 2.1 Skema Umum Sistem Informasi

Secara umum sebuah sistem komunikasi memiliki 5 komponen penting seperti yang ditunjukan pada gambar diatas. Kelima bagian sistem komunikasi [Shann-1948] adalah sebagai berikut :

- Sumber Informasi (*Information Source*).

Sumber informasi memproduksi pesan/*message* atau deret *message* yang akan sampaikan ke penerima di *destination*. Tipe-tipe *message* bisa bervariasi mulai dari (1) kumpulan abjad seperti dalam telegram atau *teletype system*, (2) sebuah fungsi waktu, *f(t)*, seperti pada radio atau telepon, (3) sebuah fungsi waktu, *f(t)*, dan variable lainnya seperti pada televisi.

Message yang ditransmisikan berupa simbol-simbol. Simbol yang dihasilkan tergantung dari probabilitas kemunculan simbol-simbol tersebut. Model matematis dari sebuah sumber informasi dimodelkan dalam himpunan probabilitas, yang lebih dikenal dengan sebutan proses stokastik (*stochastic process*). Pada dasarnya sebarang proses stokastik dihasilkan urutan simbol diskrit

terpilih dari sebuah himpunan simbol yang terhingga (*finite set*) dapat dianggap sebagai sebuah sumber diskrit (*diskrit source*) [Shann-1948].

Shannon menyatakan dalam papernya [Shann-1948] bahwa sumber informasi dibedakan menjadi 2 macam yaitu *discrete mem oryless source* (DMS) dan *markov-k source*. Sebuah source χ dengan sebuah alfabet A didefinisikan sebagai sebuah proses random (sebuah deretan variabel-variabel random x_i, i = 1,...) dalam bentuk $\chi = x_1 x_2 \ldots$, di mana tiap variabel random X_i mempunyai sebuah nilai dari alfabet A. selanjutnya, diasumsikan bahwa alfabet tersebut berisi sejumlah simbol-simbol yang terbatas (M), contohnya, $A = \{a_1, a_2, \ldots, a_M\}$.

Sebuah DMS adalah seperti simbol-simbol berkelanjutan yang independen secara statistik. Dispesifikasikan sebagai probabilitas $p(a_i) = p_i$, dimana i= 1, ..., M sehingga $p_1 + \ldots + p_M = 1$. Source yang paling realitis dapat dimodelkan lebih baik dengan proses-proses random *markov-k*. Sebuah sumber *markov-k* dapat dispesifikasikan dengan probabilitas kondisional $p(x_j = a_i | x_{j-1}, \ldots, x_{j-K})$ untuk semua j, $a_i \in A$.

Menurut teori informasi, isi informasi sebuah simbol berhubungan dengan besarnya perkiraan simbol tersebut muncul atau tidak. Jika sebuah simbol dengan probabilitas kemunculan rendah terjadi, sejumlah informasi akan dialirkan dari keberadaan simbol yang lebih mungkin muncul atau dengan kata lain simbol dengan probabilitas kemunculan rendah memiliki nilai informasi lebih tinggi dari pada simbol yang memiliki probabilitas kemunculan tinggi

- *Transmitter.*

Transmitter berfungsi mengubah *message* supaya bisa dikirim melalui *channel* komunikasi. *Transmitter* dalam telepon mengubah tekanan suara menjadi frekuensi gelombang listrik yang proporsional. *Transmitter* sering juga disebut sebagai *encoder*

- *Channel.*

Media yang digunakan untuk menyampaikan *message* dari *transmitter* ke *receiver*. *Channel* bisa berupa sepasang kabel, kabel coaxcial, gelombang frekuensi radio dan lain-lain

- *Receiver.*

Receiver bertugas untuk merekontruksi sinyal menjadi *message*, proses kebalikan dari yang dilakukan oleh *transmitter*. *Receiver* sering juga disebut dengan nama lain *decoder*

- *Destination*

Yaitu orang atau sesuatu dimana *message* tersebut akan disampaikan.

Dalam sebuah sistem komunikasi terdapat hubungan yang kritis antara elemen sistem komunikasi yaitu kekuatan/daya sumber informasi/*information source*, bandwidth media yang digunakan dalam melakukan transmisi informasi, dan *noise* yang mempengaruhi signal yang menyampaikan *message* tersebut.

Masalah mendasar yang dihadapi didalam sistem komunikasi adalah bagaimana menghasilkan kembali informasi yang sama persis dengan informasi sumber informasi asalnya atau paling tidak mendekati dengan sumber asalnya.

Hal ini disebabkan karena tidak adanya *channel* yang benar-benar sempurna terbebas dari *noise* (*noiseless channel*). Sebagai contoh jalur telepon tidak akan terbebas dari *cross-talk* dengan line telepon yang lain, radio transmitter dipengaruhi oleh radiasi permukaan bumi dan atmosfer.

Ada 2 cara yang bisa digunakan untuk mengatasi masalah ini yaitu dengan *physical solution* dan *system solution*. Solusi fisik yang digunakan untuk mengurangi *thermal noise* channel antara lain dengan memperbesar signal atau digunakan sistem pendingin untuk mengurangi panas. Secara umum solusi fisik akan meningkatkan biaya channel komunikasi.

Alternatif yang menarik ditawarkan oleh solusi sistem, dimana *noisy channel* memang tidak dapat dihindari. Yang perlu dilakukan adalah bagaimana mendeteksi dan memperbaiki *error signal* yang diterima. Pada solusi sistem memiliki kelebihan karena yang perlu dilakukan adalah menentukan komputasi tepat yang diperlukan pada sisi *transmitter* dan *receiver*.

Solusi sistem yang menarik diberikan oleh Teori Informasi dan Teori Pengkodean (*Coding Theory*). Batasan teoretis dan potensi sebuah sistem (*theoretical limitation & potential of systems*) lebih ditekankan pada Teori Informasi sedangkan *coding teory* lebih ditekankan pada pembuatan *encoding* dan *decoding* sistem

2.3 ENTROPY

Teori informasi adalah model matematis terhadap intusi manusia tentang bagaimana mengukur besar informasi yang terdapat pada suatu *message*.

Informasi adalah suatu simbol yang mengandung berita yang sama sekali baru sehingga tidak dapat diperkirakan atau diprediksi sebelumnya [Shann-1948].

Ukuran besar informasi yang terdapat dalam suatu *message* disebut sebagai nilai informasi (*Information* Content). Jika suatu *message* yang diterima dapat diperkirakan isinya sebelum *message* tersebut diterima, maka *information content* yang dimiliki oleh *message* tersebut bernilai nol. *Information content* sebuah simbol diukur dengan satuan bits dan didefinisikan sebagai berikut:

$$h(x) = \log_2 \frac{1}{p(x)} \tag{2.1}$$

Tingkat kerandoman dari suatu sumber informasi dapat digambarkan sebagai sebuah *entropy* dari sumber informasi itu sendiri. Pendekatan aksiomatik digunakan untuk menentukan besarnya *entropy* dari sebuah distribusi probabilitas P=($p1$, $p2$, ..., pn) sebagai akumulasi dari *information content* dari masing-masing komponen distribusi probabilitas P [Shann-1948]. *Entropy* didefinisikan sebagai berikut:

$$h(p) = \sum_{i=1}^{n} p_i \log \frac{1}{p_i} \tag{2.2}$$

Untuk sebuah distribusi probabilitas p dengan n kejadian (*event*), definisi di atas dihasilkan beberapa kesimpulan antara lain adalah:

- *Entropy* (h) bernilai 0, jika dan hanya jika p hanya ada satu kejadian (*event*) yang mempunyai probabilitas 1. Dimana dapat dipastikan bahwa sebelumnya peristiwa tersebut pasti benar terjadi. Hal ini sama saja bahwa kejadian tersebut bukan informasi karena dapat diprediksi sebelumnya.

- *Entropy* (h) bernilai $^2log\ n$, jika semua kejadian (*event*) mempunyai probabilitas yang sama ($1/n$). *Information content* yang dimiliki oleh masing-masing probabilitas sama besar, sehingga tidak dapat diprediksi informasi mana yang akan diterima.
- *Entropy* dari gabungan probabilitas x diikuti y, $p(x,y)$ bernilai kurang dari atau sama dengan penjumlahan *entropy* x dan *entropy* y, $h(x,y) \leq h(x)+h(y)$. Nilai *entropy* $h(x,y)$ akan sama dengan $h(x) + h(y)$, jika kedua distribusi tersebut saling independen.
- Setiap perubahan yang menuju pada persamaan nilai distribusi probabilitas p_1, p_2,...,p_n akan meningkatkan nilai *entropy* distribusi probabilitas tersebut.

Entropy kondisional (*conditional* entropy) dari y, $H_x(y)$ adalah rata-rata *entropy* y untuk setiap nilai x yang didefinisikan sebagai:

$$hx(y) = \sum_{i,j} p(i,j) \log 2 \frac{1}{p(i,j)} \tag{2.3}$$

Entropy relatif atau Kullback-Leibler Divergence (KL-Divergence) [Lin-1991] antara dua buah distribusi probabilitas $p(x)$ dan $p(y)$ adalah ukuran jarak antara dua buah distribusi probabilitas. *Entropy* relatif sering digunakan dalam model klasifikasi dengan framework teori informasi. *Entropy* relatif didefinisikan sebagai:

$$KL(p1, p2) = \sum_{x \in X} p1(x) \log \frac{p1(x)}{p2(x)} \tag{2.4}$$

Dimana x adalah variable random diskrit yang nilainya berasal dari himpunan X dengan distribusi probabilitas $p(x)$. KL-Divergence mempunyai nilai

yang selalu tidak negatif, akan tetapi nilainya dapat tidak terhingga apabila $p1(x) \neq 0$ dan $p2(x) = 0$. Jensen-Shannon Divergence (JS-Divergence) juga digunakan untuk mengukur jarak (*distance*) dari sejumlah terhingga distribusi probabilitas. Tidak seperti nilai KL-Divergence bisa bernilai tak terhingga, nilai JS-Divergence selalu bernilai terhingga. JS-Divergence didefinisikan sebagai:

$$JS_{\pi}(\{p_i : 1 \leq i \leq n\}) = H(\sum_{i=1}^{n} \pi_i p_i) - \sum_{i=1}^{n} \pi_i H(p_i) \tag{2.5}$$

Mutual information adalah ukuran jumlah informasi yang sebuah variable random mengandung variable random yang lain. Jika X, Y adalah variable random diskrit dengan distribusi probabilitas masing-masing adalah $p(x)$ dan $p(y)$ maka *Mutual Information* dari *X dan Y*, $I(X,Y)$ didefinisikan sebagai:

$$I(X;Y) = \sum_{x} \sum_{y} p(x,y) \log \frac{p(x,y)}{p(x) * p(y)} \tag{2.6}$$

BAB III
PENGGUNAAN WORD CLUSTER PADA KLASIFIKASI DOKUMEN

Pada bab ini dibahas tentang *distributional word clustering* serta peranannya dalam mengurangi jumlah *feature* klasifikasi dokumen. Pada bab ini juga dibahas tentang hibernate sebagai *object relational persistent mapping*. Sebelumnya akan dibahas mengenai klasifikasi dokumen menggunakan *naïve bayes*

3.1. REPRESENTASI DOKUMEN

Pengklasifikasian dokumen (*Document Classification*) adalah sebuah metode, yang dilakukan secara manual atau otomatis, untuk menempatkan dokumen dalam kategori berdasarkan tema (*class label*) yang ada pada dokumen tersebut [Pratt–1999].

Untuk dapat melakukan pengorganisasian pada dokumen, hal yang pertama kali dilakukan terlebih dahulu adalah merepresentasikan dokumen. Menurut Wanda M Pratt [Pratt-1999], ada beberapa cara yang digunakan untuk merepresentasikan dokumen antara lain dengan *vector space model*, *controlled vocabulary* dan *structured document*.

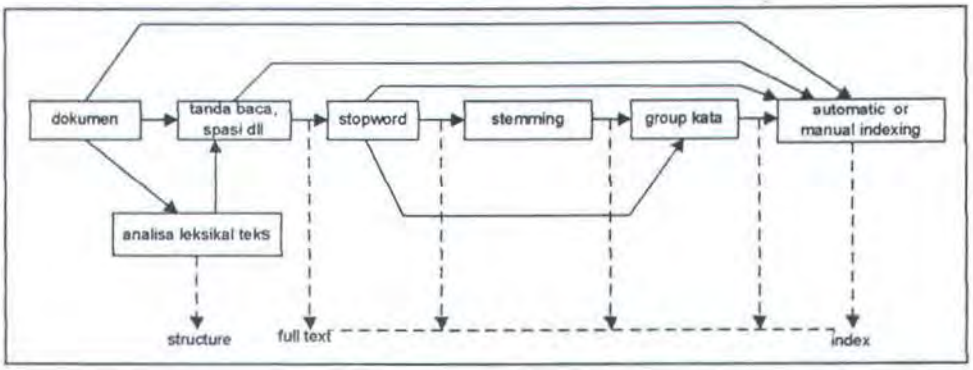
3.1.1. Vector Space Model

Model ini paling banyak dipakai untuk merepresentasikan dokumen. Dokumen digambarkan sebagai *vector of term* dari *term-term* yang terdapat pada dokumen tersebut [Pratt-1999]. Elemen ke-*i* dari vektor dokumen adalah jumlah *term i* yang terdapat dalam dokumen tersebut. Setiap vektor dokumen

diperlakukan sebagai sebuah titik dalam koordinat vektor. Dokumen dipandang sebagai sebuah titik dalam ruang vektor *term* multidimensi (*multidimensional term space*). Dari paradigma tersebut diperoleh cara pandang yang intuitif tentang bagaimana memandang dokumen berdasarkan posisinya dalam *space*, dimana kemiripan antar dokumen dihitung berdasarkan jarak antar titik tersebut.

Definisi *term* menurut Zobel dan Moffat [Zobel-1998], *term* adalah konsep yang teridentifikasi dalam dokumen. Untuk setiap dokumen teks, setiap kata didalam didalam dokumen adalah sebuah *term*, setelah kata tersebut mengalami proses pembentukan kata dasar sebelumnya.

Proses *preprocessing* dokumen perlu dilakukan terlebih dahulu sebelum mendapatkan *vector term* yang final. Langkah-langkah umum yang digunakan dalam *preprocessing* dokumen ditunjukkan pada gambar 3.1.



Gambar 3.1. Tahapan *Preprocessing* Dokumen

Menurut Baeza-Yates ada 5 tahapan yang dilakukan untuk melakukan *preprocessing* dokumen [Baeza-1999], yaitu:

1. Analisa leksikal teks.
2. Penghilangan *stopword*.
3. *Stemming*.

4. Pemilihan kata terindeks untuk menentukan grup kata.
5. konstruksi struktur kategori kata (*thesaurus*).

3.1.1.1. Analisa Leksikal Teks

Analisa leksikal adalah suatu proses transformasi teks yang masuk (dokumen) menjadi kata-kata (kata-kata kandidat yang nantinya akan di adaptasikan sebagai indeks kata). Hal yang dilakukan pada tahap ini antara lain adalah pengidentifkasian tanda spasi, penanganan digit, nomor, tanda hubung dan tanda baca dan besar kecil huruf.

" "\t\n\r\f\'\"\\1234567890!@#$%^&*()_+-={}|[]:;<,>.?/~

Gambar 3.2. Tanda Baca yang tidak disertakan dalam *indexing*

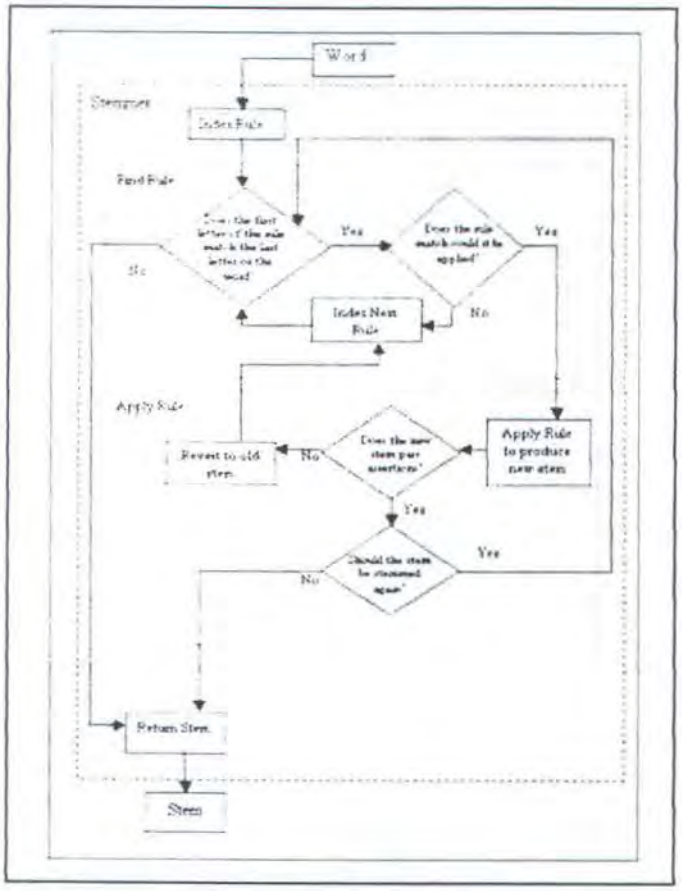
Penanganan bentuk besar-kecilnya huruf juga diperhatikan selama tahap analisa leksikal. Pada umumnya semua huruf dikonversi ke huruf besar atau huruf kecil semua. Hal ini didasarkan pada kenyataan bahwa pada umumnya besar atau kecilnya huruf tidak membedakan makna kata.

3.1.1.2. Penghilangan Stopword

Ada beberapa kata dalam bahasa inggris (seperti '*the*', '*and*', '*of*', '*to*', dan lain-lain) yang sering muncul dalam dokumen namun tidak dapat digunakan sebagai *term* pengindeks. Kata-kata yang sering muncul ini disebut sebagai *stopword* dan biasanya dihilangkan dari list indek yang potensial. Sedangkan kumpulan *stopword* disebut sebagai *stoplist*.

3.1.1.3 Stemming

Selain kedua proses diatas, juga dilakukan proses mengganti kata dengan bentuk akar morfologinya, seperti kata *compications* menjadi *complication*. Dalam banyak kasus berbagai varian kata ini memiliki interpretasi semantik yang sama dan dapat dianggap merupakan satu kesamaan. Bentuk sederhana dari metode pembentukan *morfologi* kata menjadi bentuk akar kata dikenal dengan *stemming*. *Stemming* terdiri dari dua macam yaitu 1) mengganti kata jamak menjadi kata tunggal dan 2) menghilangkan *suffiks* (seperti *quickly, quicker* menjadi *quick*).



Gambar 3.3. Alur kerja Paice/Husk Stemmer

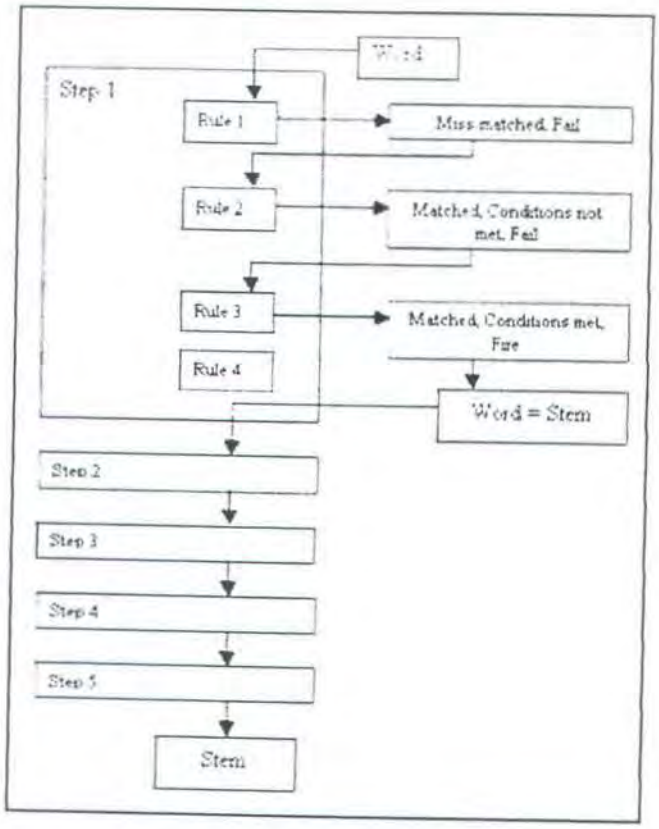
Ada beberapa macam algoritma *stemming* yang dikembangkan oleh para ahli antara lain *Paice/Husk Stemming, Porter Stemming, Lovins Stemming, Dawson Stemming, Krovets Stemming*.

Paice/Husk Stemmer adalah sebuah metode *stemming* yang dikembangkan oleh Chris Paice pada tahun 1990 di Universitas Lancaster. *Stemmer* ini adalah *stemmer* yang didasarkan pada *Iterative Stemmer* [Paice-1990]. Gambar alur kerja *Paice/Husk Stemmer* dapat dilihat pada gambar 3.3.

Pada *Paice/Husk Stemmer* digunakan sebuah file aturan (*rule file*) yang *load* dalam sebuah *list*. Sebuah indek yang dibangun untuk *list* tersebut, diurutkan berdasarkan pada huruf pertama *rule* (huruf terakhir dari kata yang akan di-*stem*). Indek ini merujuk pada elemen awal yang dimulai dengan huruf tersebut. Semua aturan disusun secara alphabet berdasarkan pada huruf awal elemen.

Porter Stemmer dikembangkan oleh Martin Porter di Universitas Cambridge pada tahun 1980. *Stemmer* ini dikembangkan pada dasar bahwa *suffixes* dalam bahasa inggris (sekitar 1200 buah) disusun berdasarkan kombinasi suffixes yang kecil dan sederhana. *Stemmer* ini merupakan jenis *linear step stemmer* [Paice-1990].

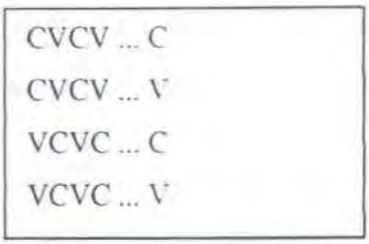
Secara spesifik algoritma ini memiliki lima *step* dimana setiap *step* memiliki beberapa *rule*. Pada setiap *step*, jika sebuah *suffix rule* sesuai dengan sebuah huruf maka *suffix* huruf tersebut akan dihilangkan. Proses ini dilakukan pada setiap *step*, sehingga hasil *stemming* baru akan diketahui setelah melalui kelima tahapan *step* tersebut.



Gambar 3.4. Alur Kerja *Porter Stemmer*

Terminologi yang digunakan Algoritma *Porter Stemmer* sebagai berikut:

- Konsonan (*consonant*)dalam sebuah kata adalah huruf-huruf selain huruf A, I, U, E, O serta huruf Y yang sebelumnya diawali dengan konsonan. Konsonan yang terdapat dalam kata TOY adalah T dan Y, sedangkan konsonan dari kata SYZYGY adalah S, Z, G. huruf selain konsonan adalah vokal (*vowel*).
- Sebuah *consonant* dinotasikan dengan *c*, dan *vowel* dinotasikan dengan *v*. sebuah *list consonant* dengan panjang lebih dari 0 dinotasikan dengan *C*, demikian juga dengan *vowel* dinotasikan dengan *V*. Sebarang kata atau bagian dari sebuah kata memiliki empat bentuk yaitu:



Gambar 3.5. Bentuk Kata Berdasarkan *consonant* dan *vowel*

Keempat bentuk kata tersebut dapat disingkat dalam bentuk ringkas [C] VCVC ... [V]. (VC){m} untuk ditunjukkan bahwa VC diulangi sebanyak m kali. Sehingga bentuk diatas dapat ditulis sebagai [C] (VC){m} [V]. *m* dinotasikan sebagai *measure* (ukuran) dari kata atau bagian kata. Nilai $m = 0$ jika kata tersebut adalah sebuah *null word*. Contoh :

$m = 0$ TR, EE, TREE, Y, BY

$m = 1$ TROUBLE, OATS, TREES, IVY

$m = 2$ TROUBLES, PRIVATE, OATEN, ORRERY

- *Suffix* dihilangkan dengan aturan sebagai berikut:

(*condition*) S1 → S2 (3.1)

Jika sebuah kata berakhir dengan *suffix* S1 dan *stem* sebelum sesuai dengan *condition*, maka S1 akan diganti dengan S2. *condition* biasanya ditunjukan dengan kontek *m*. contoh:

(*m* > *1*) EMENT → dimana S1 = "EMENT" dan S2 = *null*, aturan ini akan merubah kata REPLACEMENT menjadi REPLAC, dimana REPLAC adalah bagian dari kata dengan $m = 2$.

Bagian *condition* juga bisa berupa :

- *S -- *stem* yang berakhiran dengan huruf S (dan sama untuk huruf yang lainnya).
- **v** -- *stem* yang memiliki sebuah *vowel*.
- **d* -- *stem* yang berakhiran dengan konsonan ganda (contoh. –TT, -SS).
- *o -- *stem* yang berakhiran *cvc*, dimana *c* yang kedua bukan W, X, atau Y (contoh. -WIL, -HOP).
- Selain itu bagian *condition* juga bisa berupa *expression* dengan "and", "or", dan "not" dimana (*m*>1 and (*S or *T) untuk menguji jika sebuah *stem* memiliki *m* > 1 dan berakhiran S atau T.

Berikut ini merupakan langkah-langkah untuk menghilangkan *suffixs*:

- ❑ Step 1, Terkait dengan bentuk kata jamak (*prural*) dan kata lampau (*past participle*). Step 1 terdiri dari tiga buah bagian yaitu:

a. Step 1a.

Table 3.1 Rule Step 1a Porter Stemmer

No	Rule	Contoh
1.	SSES → SS	caresses → caress
2.	IES → I	ponies → poni ties → ti
3.	SS → SS	caress → caress
4.	S →	cats → cat

b. Step 1b.

Table 3.2 Rule step 1b Porter Stemmer

No	Rule	Contoh
1.	(*m*>0) EED → EE	feed → feed
2.	(**v**) ED →	plastered → plaster
3.	(**v**) ING →	motoring → motor

Jika *rule* yang kedua dan ketiga dilakukan maka *rule-rule* berikut ini akan dilakukan.

Table 3.3 Rule tambahan step 1b Porter Stemmer

No	Rule	Contoh
1.	AT → ATE	conflate(ed) → conflate
2.	BL → BLE	trouble(ed) → trouble
3.	IZ → IZE	siz(ed) → size
4.	(*d and not (*L or *S or *Z)) → single letter	hopp(ing) → hop tann(ed) → tan fall(ing) → fall
5.	(*m* = 1 and *o) → E	fail(ing) → fail fil(ing) → file

c. Step 1c.

Table 3.4 Rule step 1c

No	Rule	Contoh
1.	(*v*) Y → I	happy → happi

- Step 2, terdiri dari 20 *rule*.

Table 3.5 Daftar rule step 2 Porter Stemmer

No	Rule	Contoh
1.	(*m* > 0) ATIONAL → ATE	relational → relate
2.	(*m* > 0) TIONAL → TION	conditional → condition
3.	(*m* > 0) ENCI → ENCE	valenci → valence
4.	(*m* > 0) ANCI → ANCE	hesitanci → hesitance
5.	(*m* > 0) IZER → IZE	digitizer → digitize
6.	(*m* > 0) ABLI → ABLE	conformabli→ conformable
7.	(*m* > 0) ALLI → AL	radicalli → radical
8.	(*m* > 0) ENTLI → ENT	differentli → different
9.	(*m* > 0) ELI → E	vileli → vile

10.	$(m > 0)$ OUSLI → OUS	analogousli → analogous
11.	$(m > 0)$ IZATION → IZE	vietnamization → vietnamize
12.	$(m > 0)$ ATION → ATE	predication → predicate
13.	$(m > 0)$ ATOR → ATE	operator → operate
14.	$(m > 0)$ ALISM → AL	feudalism → feudal
15.	$(m > 0)$ IVENESS → IVE	decisiveness → decisive
16.	$(m > 0)$ FULNESS → FUL	hopefulness → hopeful
17.	$(m > 0)$ OUSNESS → OUS	callousness → callous
18.	$(m > 0)$ ALITI → AL	formaliti → formal
19.	$(m > 0)$ IVITI → IVE	sensitiviti → sensitive
20.	$(m > 0)$ BILITI → BLE	sensibiliti → sensible

- Step 3

Table 3.6 Daftar rule step 3 Porter Stemmer

No	Rule	Contoh
1.	$(m > 0)$ ICATE → IC	triplicate → triplic
2.	$(m > 0)$ ATIVE →	formative → form
3.	$(m > 0)$ ALIZE → AL	formalize → formal
4.	$(m > 0)$ ICITI → IC	electriciti → electric
5.	$(m > 0)$ ICAL → IC	electrical → electric
6.	$(m > 0)$ FUL →	hopeful → hope
7.	$(m > 0)$ NESS →	goodness → good

- Step 4

Table 3.7 Daftar rule step 4 Porter Stemmer

No	Rule	Contoh
1.	$(m > 1)$ AL →	revival → reviv
2.	$(m > 1)$ ANCE →	allowance → allow
3.	$(m > 1)$ ENCE →	inference → infer
4.	$(m > 1)$ ER →	airliner → airlin

5.	(*m* > 1) IC →	gyroscopic → gyroscop
6.	(*m* > 1) ABLE →	adjustable → adjust
7.	(*m* > 1) IBLE →	defensible → defens
8.	(*m* > 1) ANT →	irritant → irrit
9.	(*m* > 1) EMENT →	replacement → replac
10.	(*m* > 1) MENT →	adjustment → adjust
11.	(*m* > 1) ENT →	dependent → depend
12.	(*m* > 1 and (*S or *T)) ION →	adoption → adopt
13.	(*m* > 1) OU →	homologou → homolog
14.	(*m* > 1) ISM →	communism → commun
15.	(*m* > 1) ATE →	activate → activ
16.	(*m* > 1) ITI →	angulariti → angular
17	(*m* > 1) OUS →	homologous → homolog
18.	(*m* > 1) IVE →	effective → effect
19.	(*m* > 1) IZE	bowdlerize → bowdler

- Step 5, step ini dibagi menjadi dua bagian sebagai berikut:
  - ✓ Step 5a

Table 3.8 Daftar rule step 5a Porter Stemmer

No	Rule	Contoh
1.	(*m* > 1) E →	probate → probat
2.	(*m* = 1 and not *o) E →	cease → ceas

  - ✓ Step 5b

Table 3.9 Daftar rule step 5b Porter Stemer

No	Rule	Contoh
1.	(*m* > 1 and *d and *L) → single letter	controll → control roll → roll

Stemming tidak dilakukan algoritma porter pada kata yang panjang *stem*-nya (*m*) pendek. Seperti yang ditunjukkan pada contoh tabel dibawah, pada list B *suffix* -ATE akan dihilangkan sedangkan pada list A tidak. Hal ini ditunjukan bahwa pasangan kata DERIVATE/DERIVE, ACTIVATE/ACTIVE, DEMONSTRATE/DEMONSTRABLE, NECESSITATE/NECESSITOUS akan di stem menjadi kata yang sama.

Table 3.6 Perbedaan Stemming berdasarkan panjang *stem* (*m*)

List A	List B
RELATE	DERIVATE
PROBATE	ACTIVATE
CONFLATE	DEMONSTRATE
PIRATE	NECESSITATE
PRELATE	RENOVATE

Lovins Stemmer adalah jenis *single pass context sensitive stemmer*. *Stemmer* ini dikembangkan oleh Julia Beth Lovins dari Massachusetts Institute of Technology pada tahun 1968 [Paice-1990]. Rule yang digunakan dalam *stemmer* ini diproses dan dihasilkan dari sample kata. Selain itu *stemmer* ini juga juga mempunyai masalah pada proses pembentukan kembali kata (*reformation of word*), karena proses ini digunakan *recoding rule* untuk membentuk *stem* menjadi kata yang sesuai dengan *stem* kata lain yang memiliki makna yang sama. Proses *recoding rule* yang dilakukan tidak reliable dan sering terjadi kegagalan dalam membentuk kata dari *stem* atau menyesuaikan dengan *stem* yang memiliki makna yang sama selain itu *Lovins Stemmer* hanya dapat menghilangkan satu *suffix* kata.

Dawson Stemmer dikembangkan oleh J.L Dawson dari Literary and Linguistic Computing Centre, Cambridge. *Dawson Stemmer* merupakan stemmer *linguistic* yang komplek. Stemmer ini didasarkan pada *Lovins Stemmer* dengan menambah *suffix rule* menjadi sekita 1200 aturan, serta menghilangkan tahap *recoding rule* dengan menggantikannya dengan prosedur *partial maching* yang juga didefinisikan dalam paper *Lovins Stemmer.*

Krovetz Stemmer dikembangkan oleh Robert Krovetz dari University of Massachusetts pada tahung 1993. Stemmer ini merupakan jenis *Linguistic Morphological Inflection Stemmer* serta stemmer yang sangat kompleks [Paice-1990].

Morfologi adalah ilmu yang membahas struktur internal dari kata. Morfologi dibedakan menjadi dua bagian yaitu *inflectional* dan *derivational*. Pada *Inflectional Morphology*, prediksi perubahan sebuah kata adalah sebagai hasil dari sintak (bentuk *prural* dan *possessive* untuk kata benda, *past tense* dan *progressive* untuk kata kerja merupakan hal yang umum terjadi pada bahasa inggris). Perubahan ini tidak mempunyai efek pada kata yang berupa *'part-of-speech'* (sebuah kata benda masih tetap berupa kata benda meskipun berubah menjadi bentuk jamak). Hal sebaliknya terjadi pada *derivational morphology* dimana *suffix* bisa saja, tetapi tidak selalu, merubah arti kata. Morfologi sedikit digunakan pada bahasa inggris, sedangkan bahasa hunggaria dan yahudi memakai banyak morfologi.

Krovetz Stemmer sangat efektif untuk menghilangkan *inflectional suffixes*. Algoritma ini menghilangkan *suffix* dalam tiga langkah yaitu konversi bentuk kata

jamak menjadi kata tunggal (contoh. '-ies, '-es', '-s'), konversi dari bentuk kata lampau (*past tense*) (contoh. '-ed') menjadi bentuk kata sekarang (*present tense*), dan penghilangan '-ing'.

Sebuah studi yang dilakukan oleh Hersh dan Greenes [Pratt-1999], *stemming* yang dilakukan pada *medical information retrieval system* diperoleh kesimpulan bahwa *stemming* telah mengurangi tingkat keakurasian *searching* secara dramatis. Hal ini disebabkan karena dalam dunia kedokteran *suffixes* memiliki peranan yang vital dalam menentukan arti sebuah kata. Contoh kata *sinus* memiliki arti rongga yang berada di dalam tulang, sedangkan *sinusitis* adalah peradangan rongga hidung, dan *sinusoid* adalah diameter pembuluh kapiler yang lebar. Jika semua kata ini di-*stemming* menjadi sinus, meskipun artinya berbeda, maka hal ini akan mengurangi tingkat keakurasiannya.

Setelah mengalami proses *preprocessing*, setiap dokumen direpresentasisikan sebagai sebuah vektor *term*. Jika sekumpulan dokumen hasil memiliki *n term* yang unik maka dokumen tersebut direpresentasikan sebagai sebuah vektor dengan panjang vektor n:(t_1, t_2, ...,t_n) dimana t_i mempunyai nilai 0 jika *term i* tidak terdapat disebuah dokumen dan memiliki nilai positif jika *term i* ada. Nilai positif eksak tergantung pada skema pembobotan *term* (*term-weighting*) yang digunakan. Untuk kasus yang sederhana, nilai 1 digunakan jika *term* tersebut muncul dan 0 jika *term* tersebut tidak muncul.

3.1.2. Controlled Vocabulary

Term-term yang terdapat pada sebuah ***controlled vocabulary*** digunakan sebagai represntasi dokumen. *Controlled Vocabulary* adalah sekumpulan *term*

yang telah di definisikan sebelumnya. Semantic And Probabilistic Heuristic Information Retrieval Environment (SAPHIRE) adalah salah satu contoh sistem informasi temu kembali (IR System) untuk domain kedokteran yang memakai *controlled vocabulary* [Pratt-1999]. Pemetaan (*mapping*) kata dan frasa yang dilakukan dalam sebuah dokumen dalam bentuk sekumpulan *term* (*canonical terms*). Daftar *canonical term* berasal dari UMLS Metathesaurus, sebuah terminology model yang dikhususkan untuk dunia kedokteran

Dokumen direpresentasikan dalam SAPHIRE dengan pembobotan vektor dari *canonical term* ini. Beberapa studi yang telah dilakukan untuk membandingkan *controlled vocabulary* dengan sistem *vector space* menunjukkan bahwa performa sistem yang menggunakan *controlled vocabulary* tergantung pada besarnya kualitas model terminologi yang digunakan [Pratt-1999].

3.1.3. Structured Document

Dokumen dapat distrukturisasi berdasarkan perbedaan komponen *syntactic*, seperti yang dijelaskan pada bagian 3.1.3.1 ataupun didasarkan komponen *semantic*, seperti yang dijelaskan pada bagian 3.1.3.2 dan 3.1.3.3

3.1.3.1. Document Component

Dokumen direpresentasikan dalam bentuk komponen *syntactic* seperti judul, pengarang, jurnal, *keyword* dan abstrak. Seperti halnya yang terdapat dalam model *vector space*, dokumen direpresentasikan dengan kata-kata yang terdapat dalam dokumen tersebut, sehingga setiap komponen memiliki vektor word tersendiri. Pencarian dilakukan eksplisit kemunculan kata-kata dalam komponen tertentu seperti judu, pengarang, jurnal.

Salah satu alternatif pendekatan yang digunakan adalah merepresentasikan secara eksplisit komponen semantic dari sebuah dokumen dari pada hanya menampilkan komponen sintaktiksnya. RiboWeb memakai pendekatan ini untuk merepresentasikan publikasi jurnal ilmiah yang terkait dengan ribosome [Pratt-1999]. RiboWeb memiliki *knowledge base* dari data-data yang relevan yang telah dipublikasikan dan sebuah modul komputasi yang mampu mengolah data-data ini untuk digunakan untuk pengujian hipotesis terhadap struktur ribosome [Pratt-1999].

3.1.3.2. Structured Abstracts

Struktur semantic dan format yang spesifik terhadap sebuah abstrak dokumen ditekankan pada ***Structured Abstracts***. Banyak sekali journal yang mengadopsi abstrak terstruktur untuk membantu pembaca mengakses dokumen dan meningkatan kecepatan pencarian elektronik [Pratt-1999].

Masalah yang muncul pada abstrak terstruktur terutama pada terbatasnya kemampuan mereka untuk dipertemukan tujuan-tujuan terstruktur mereka. Pertama, setiap jurnal memiliki aturan dan format tersendiri untuk struktur dokumen. Bervariasinya struktur ini mempersulit *search system* dalam menyediakan akses yang seragam ke struktur dokumen ketika dilakukan pencarian terhadap berbagai jurnal [Pratt-1999].

3.1.3.3. Context Model

Context Model menyediakan struktur semantik untuk dokumen teks *(full-text document)* tanpa ditekankan format aturan yang spesifik [Pratt-1999]. Setiap kalimat dalam sebuah document dikaitkan kedalam satu atau lebih konteks yang

menggambarkan tema semantik dari kalimat tersebut. Contoh, pada sebuah *search system* dapat dibedakan sebuah dokumen yang terdapat *term "breast cancer"* dalam konteks yang memenuhi kriteria dari sebuah studi dengan *term "breast cancer"* dalam konteks efek yang merugikan dari sebuah intervensi. Dengan demikian dapat dilakukan pencarian berdasarkan *term* yang dispesifikasikan dan konteks dimana term tersebut seharusnya muncul.

Purcell [Pratt-1999] telah membangun sebuah *context model* untuk artikel riset klinik, *case report*, dan artikel *review* dalam literatur kedokteran. Purcell menunjukkan bahwa *context model* dapat dihasilkan presisi yang lebih baik daripada memakai pencarian yang sama dengan Boolean, sebuah *full-text search system*. Konteks dikaitkan secara manual terhadap individual kalimat, sehingga mereka dapat digunakan hanya pada setting penelitian saja.

3.1.4. NAÏVE BAYES CLASSIFIER

Salah satu metode pembelajaran praktis yang sering digunakan dalam klasifikasi dokumen teks adalah metode pembelajaran naïve bayes (*naïve bayes learner*), atau yang sering dikenal sebagai *naïve bayes classifier*. Naïve bayes memakai metode pembelajaran dimana setiap *instance x* dideskripsikan sebagai sebuah konjungsi dari nilai atribut (*attribute value*) a dan fungsi target (*target function*) $f(x)$ dapat diambil dari sebarang nilai dari beberapa himpunan terhingga v [Mitch-1997].

Pendekatan probabilistik digunakan dalam *bayesian learning* untuk menarik suatu kesimpulan. Hal ini didasarkan pada asumsi yang menyatakan bahwa kuantitas suatu minat diatur oleh distribusi probabilitas serta keputusan

optimal yang dapat diambil dapat digunakan gabungan antara distribusi probabilitas ini dengan data yang diamati [Mitch-1997].

Metode pembelajaran bayes sangat relevan dalam bidang *machine learning*. Beberapa alasan yang bisa digunakan adalah (1) *bayesian learning* memakai probabilitas secara eksplisit dan (2) *bayesian learning* memakai perspektif yang berbeda untuk memahami berbagai algoritma yang secara eksplisit memanipulasi probabilitas. Standar keputusan optimum yang optimum diperoleh dengan menggunakan metode Bayesian dibandingkan dengan metode praktis lain yang dapat diukur [Mitch-1997]. Secara umum fitur-fitur yang dimiliki oleh *bayesian learning* seperti yang ditunjukkan pada tabel 3.11.

Tabel 3.11 Fitur-fitur *Bayesian Learning*

No	Fitur
1.	Setiap contoh *training* yang digunakan dapat secara *incremental* meningkatkan atau menurunkan probabilitas estimasi kebenaran sebuah hipotesa
2.	*Prior knowledge* dapat dikombinasikan dengan data yang diamati untuk menentukan probabilitas sebuah hipotesa. Dalam *bayesian learning*, *prior knowledge* diberikan *prior probability* untuk setiap kandidat hipotesa dan distribusi probabilitas pada data yang diamati untuk setiap hipotesa yang dimungkinkan
3.	Dapat mengakomodasikan hipotesa digunakan prediksi probabilitas (contoh. Hipotesa 93% pasien yang mengidap pneumonia mempunyai kesempatan untuk sembuh total)
4.	*Instance* baru dapat diklasifikasikan dengan mengkombinasikan prediksi dari beberapa hipotesis dengan pembobotan probabilitas yang dimilikinya

Salah satu kesulitan didalam mengaplikasikan metode bayesian adalah menentukan pengetahuan awal (*prior knowledge*) dari berbagai probabilitas. Pengetahuan awal ini sering kali di-*estimasi* dari beberapa faktor seperti *background knowledge*, data yang tersedia sebelumnya dan asumsi tentang bentuk distribusi probabilitas.Teorema Bayes didefinisikan sebagai berikut:

$$P(h \mid D) = \frac{P(D \mid h) * P(h)}{P(D)} \qquad (3.8)$$

Dimana :

- P(h) *Prior probability* dari h, *background knowledge* yang ketahui bahwa h benar.
- P(D) *Prior probability* dari training data D yang diamati
- P($D|h$) Probabilitas training data D yang diamati bila hipotesa h diketahui
- P($h|D$) *Posterior probability*, adalah probabilitas h jika training data D diketahui

Cara utama yang diberikan teorema bayes untuk menghitung *posterior probability* untuk setiap *training hypothesis* dengan diberikan sejumlah data training. Basis untuk algoritma pembelajaran (*learning algorithm*) dihitung secara langsung untuk menghitung probabilitas setiap hipotesa dihasilkan keluaran yang paling mungkin (most probable) [Mitch-1997].

3.1.4.1. Klasifikasi Dokumen Dengan Naïve Bayes

Hasil yang cukup baik diperoleh klasifikasi dokumen dengan menggunakan berbagai varian metode bayesian [Pazza-1997]. Pendekatan probabilistik diperlukan asumsi mengenai bagaimana menghasilkan data dan membentuk model probabilitas berdasarkan asumsi itu. Estimasi parameter dari

generative model dilakukan dengan menggabungkan model probabilitas dengan sejumlah *data training* yang sudah terlabeli.

Naïve Bayes adalah salah satu metode bayesian yang paling sederhana diantara metode *bayesian learning* yang lainnya. Model yang saling independen terhadap *class* adalah asumsi dasar *naïve bayes*. Meskipun asumsi ini tidak ada dalam dunia nyata, akan tetapi hasil klasifikasi yang baik bisa diperoleh dengan *naïve bayes* [McCall-1998].

Ada dua macam model yang digunakan dalam melakukan klasifikasi dokumen teks dengan menggunakan metode *naïve bayes* [McCall-1998], kedua model tersebut dibedakan menurut representasi vektor dokumen dalam klasifikasi. Yang pertama adalah digunakan atribut vektor biner yang menunjukkan suatu kata ada atau tidak dalam sebuah dokumen. Jika suatu kata muncul dalam dokumen maka vektornya bernilai 1, sedangkan jika kata tersebut tidak ada maka vektornya bernilai 0. Jumlah kata yang muncul dalam dokumen tidak digunakan, dengan kata lain dokumen dapat dianggap sebagai sebuah kejadian (*event*) dan ada tidaknya kata adalah atribut dari kejadian tersebut. Model ini sering disebut dengan *multi-variate bernoulli event model*.

Sedangkan pada model kedua, dokumen digambarkan sebagai representasi dari himpunan kata-kata yang ada didalamnya. Urutan kata-kata diabaikan, dan jumlah kata-kata yang terdapat pada dokumen tersebut yang dipakai. Dalam hal ini setiap kata adalah kejadian (*event*) dan dokumen adalah sekumpulan dari *word event*. Model ini dinamakan sebagai *multinomial event model*.

Riset yang dilakukan oleh McCallum dan Nigam [McCall-1998] menunjukkan bahwa klasifikasi *naïve bayes* dengan *multinomial event model* lebih baik daripada klasifikasi *naïve bayes* yang menggunakan *multi-variate bernoulli event model*. *Naïve bayes* dengan *multinomial event model* digunakan dalam tugas akhir ini.

$$p(c_i \mid d) = \frac{p(d \mid c_i)p(c_i)}{p(d)} \tag{3.9}$$

Diberikan C = {c_1, c_2, . . .,c_i}, himpunan kelas/kategori dokumen, dan W = {w_1, w_2, . . ., w_t} adalah himpunan word (feature) yang terdapat dalam kelas tersebut. Probabilitas dokumen d termasuk dalam kelas c_i, $p(c_i|d)$ didefinisikan pada gambar 3.13. Untuk menghitung Kategori yang paling mungkin untuk sebuah dokumen digunakan rumus berikut :

$$c^*(d) = \arg\max_c (p(c_i \mid d) = p(ci) \prod_{t=1}^{m} p(wt \mid ci)^{n(wt,d)}) \tag{3.10}$$

Dimana:

$n(w_t,d)$ Jumlah kemunculan word w_t dalam dokumen d

$p(c_i)$ *Class prior* diukur dengan menggunakan *maximum likehood estimate* yaitu: $p(ci) = \frac{|c|}{\sum_j |cj|}$

3.2. DIVISIVE FEATURE CLUSTERING

Tidak semua *term* yang dihasilkan dalam *preprocessing* bisa langsung digunakan sebagai karena jika semua *term* digunakan langsung dalam klasifikasi maka akan mempertinggi waktu komputasi. Masalah utama yang dihadapi oleh

algoritma klasifikasi adalah tingginya dimensi dari *feature space* [Yang-1997]. *Feature space* yang dihasilkan pada tahap *preprocessing* terdiri dari semua *term* unik yang terdapat dalam dokumen, yang bisa terdiri dari puluhan atau ratusan ribu *term*.

Secara umum ada dua macam pendekatan yang digunakan untuk melakukan pengurangan jumlah *term/feature* [Sloni-2001], yaitu *feature selection* dan *feature clustering*. Prosedur standar sering yang digunakan untuk mengurangi jumlah *feature* adalah *feature selection*.

3.2.1. Feature Selection

Feature selection adalah metode standar yang digunakan sering digunakan dalam *Information Retrieval* didalam melakukan pengurangan jumlah *feature*. Cara ini dilakukan dengan memilih sebagian dari *feature* yang ada. *feature* yang telah dipilih tadi sebagai *feature* yang digunakan dalam melakukan klasifikasi [Sloni-2001]. Menurut Yiming Yang & Jan O. Pedersen [Yang-1997] *feature selection* adalah menghilangkan *non-informative terms* menurut *corpus statistics* dan melakukan kontruksi *feature* baru dengan cara mengkombinasikan *lower level feature* (*term*) ke dalam dimensi orthogonal yang lebih tinggi levelnya. Ada beberapa kriteria yang digunakan untuk memilih *feature* [Yang-1997] antara lain adalah: *document frequency* (DF), *information gain* (IG), *mutual information* (MI), χ^2 *statistic* (CHI), dan *term strength* (TS).

Document frequency (DF) adalah jumlah dokumen dimana sebuah *term T* muncul. Frekuensi dokumen untuk setiap *term* yang muncul dalam *training corpus* dihitung. *Term* yang frekuensi kemunculan dokumen kurang dari batasan

yang telah ditetapkan dihilangkan dari *feature space*. Asumsi dasar yang digunakan dalam *document frequency* adalah *feature* yang jarang muncul dalam dokumen selain tidak informatif, juga tidak mempengaruhi kinerja klasifikasi secara global. DF tidak dipakai dalam mengurangi jumlah *feature* secara agresif karena asumsi umum yang digunakan dalam *information retrieval* bahwa *term* yang memiliki DF kecil adalah *term* yang *informative* sehingga sebaiknya tidak dihilangkan [Yang-1997].

$$G(t) = -\sum_{i=1}^{m} \Pr(c_i) \log \Pr(c_i) + \Pr(t) \sum_{i=1}^{m} \Pr(c_i \mid t) \log \Pr(c_i \mid t) + \Pr(\bar{t}) \sum_{i=1}^{m} Pr(c_i \mid \bar{t}) \log \Pr(c_i \mid \bar{t}) \tag{3.2}$$

Information Gain (IG) mengukur jumlah bits informasi yang diperoleh dengan cara mengetahui *presence* atau *absence term* dalam sebuah dokumen [Yang-1997]. Definisi *Information Gain* diberikan pada gambar diatas dimana $\{c_i\}_{i=1}^{m}$ adalah kategori dalam klasifikasi. *Term* yang nilainya dibawah batasan yang telah ditetapkan sebelumnya dihilangkan dalam klasifikasi.

Mutual Information (MI) adalah kriteria yang secara umum digunakan dalam bahasa pemodelan statistik dari *word association*. MI antara *term t* dengan kategori *c* didefinisikan sebagai :

$$I(t,c) = \log \frac{\Pr(t \wedge c)}{\Pr(t) x \Pr(c)} \tag{3.3}$$

Nilai I sama dengan 0 jika *t* dan *c* adalah independen. Kelemahan dari MI adalah nilainya sangat dipengaruhi oleh probabilitas marginal *term* [Yang-1997].

χ^2 *statistic* (CHI) mengukur *lack of independence* antara *t* dan *c* dan bisa dibandingkan dengan distribusi χ^2 dengan 1 derajat kebebasan [Yang-1997]. Sama seperti MI, χ^2 *statistic* juga memiliki nilai natural 0 jika *t* dan *c* independen. Perbedaan antara CHI dengan MI adalah χ^2 adalah *normalized value* sehingga nilai χ^2 dapat dibandingkan dengan *term* dalam kategori yang sama.

Term Strength (TS) melakukan estimasi *term importance* berdasarkan pada bagaimana secara umum sebuah *term* akan muncul pada dokumen-dokumen yang saling terkait [Yang-1997]. Sejumlah dokumen digunakan sebagai *training set* untuk mendapatkan pasangan dokumen dimana nilai kemiripannya diatas ambang batas yang ditetapkan sebelumnya. Untuk mengukur kemiripan antar dokumen digunakan nilai cosinus dari 2 vektor dokumen. Jika x dan y adalah pasangan dokumen maka TS diukur berdasarkan estimasi probabilitas kondisional kemunculan *term t* pada y dengan kemunculan *term t* pada dokumen x, yang dirumuskan sebagai

$$s(t) = \Pr(t \in y \mid t \in x) \tag{3.4}$$

3.2.2. Feature Clustering

Feature clustering adalah cara alternatif yang bisa digunakan untuk mengurangi dimensi *feature* [Perei-1993][Baker-1998][Sloni-2001][Dhill-2003]. Pada *Feature clustering* dilakukan pengelompokan *feature* yang "sama" kedalam sejumlah klaster, dan menggunakan klaster sebagai *feature*. Riset yang dilakukan oleh Baker dan Mc Callum [Baker-1998], Slonim dan Tishby [Sloni-2001] menunjukkan bahwa *feature clustering* lebih efektif daripada *feature selection* [Yang-1997], khususnya pada jumlah *feature training* yang sedikit. Menurut

Douglas Baker [Baker-1998], *feature clustering* memiliki keunggulan dalam mengurangi jumlah *feature* yang redundan sedangkan *feature selection* memiliki keunggulan di dalam menghilangkan *dentrimental, noisy features.*

Ada beberapa macam cara yang digunakan didalam melakukan klastering yaitu diantaranya hierarchical clustering dan k-means clustering disamping teknik klastering yang lainnya seperti minimum spanning tree (MST), mutual neighborhood, single-link, complete-link.

Sebuah metode yang digunakan untuk mengukur kemiripan *word* dan menggabungkan *word* yang memiliki kemiripan sama kedalam satu klaster yang sama pada algoritma *word clustering*. Hal ini dilakukan dengan mengukur jarak antar 2 titik klaster (*pairwise clustering*) atau mengukur distorsi antara sebuah *data point* dengan sebuah *class centroid* (*vector quantization*). *Distance matrix* atau *distorsion measure* digunakan sebagai metode pengukuran yang dapat diadaptasikan dalam berbagai cara untuk mendapatkan sejumlah kelas dengan *intraclass distorsion* yang rendah atau *high intraclass connectivity.*

Pada algoritma *K-means clustering* ditentukan sebanyak k klaster dan distribusi diantara k klaster tersebut dengan digunakan kesamaan *centroid* untuk memperoleh klaster. Nilai *Initial cluster center* dan nilai k diperlukan oleh *K-means clustering* sebagai parameter tambahan untuk membentuk klaster baru, menggabungkan klaster yang sudah ada dan melakukan pendeteksian tepi [Jain-1999]. *Entropy* dan *F Measure* adalah 2 skema yang populer digunakan untuk mengukur kualitas dari klaster yang dihasilkan.

Ada 3 keuntungan apabila digunakan *feature clustering* [Baker-1998] yaitu:

1) Dihasilkan *useful semantic word clustering*. *Feature clustering* secara tidak langsung dapat dihasilkan grup kata (klaster) yang secara semantik saling terkait. Efeknya adalah secara otomatis menghasilkan *thesaurus*.
2) Keakurasian klasifikasi.
3) Model klasifikasi yang kecil. Dengan memakai klaster, maka sejumlah parameter klasifikasi yang terpisah untuk banyak kata dapat diganti dengan sebuah parameter untuk sebuah *word cluster*.

Point 1 dan 2 merupakan manfaat yang juga diperoleh jika menggunakan *feature selection* untuk mengurangi dimensionalitas. Sedangkan point 3 merupakan manfaat yang diperoleh jika menggunakan *feature clustering*.

Distributional word clustering adalah *feature clustering* dengan menggabungkan *word* yang mempunyai kesamaan distribusi probabilitas dalam satu klaster [Baker-1998]. Kemiripan *feature* diukur berdasarkan kesamaan distribusi probabilitas dengan menggunakan berbagai variasi entropy relatif, yang lebih dikenal dengan *Kullback-Leibler Divergence* (KL-*Divergence*). Didalam konteks teori informasi, KL-*Divergence* mengukur efisien atau tidaknya sebuah *message* yang dihasilkan dengan menggunakan distribusi probabilitas p_1 di-*encode* untuk di transmisikan dengan code yang menggunakan distribusi p_2.

Ada beberapa prosedur yang digunakan untuk membentuk klaster berdasarkan distribusi probabilitas *feature*, framework *Information Bottleneck* (IB) digunakan oleh Noam Slonim & Naftali Tishby [Sloni-2001] sedangkan framework Teori Informasi digunakan oleh Dhillon dkk [Dhill-2003].

Jika sebuah variable random C adalah variable random untuk *class labels* $\{C_1, C_2,...,C_m\}$ dan diberikan sebuah *word* w_t pada *class labels*, maka distribusi probabilitas w_t didefinisikan sebagai $P(C|w_t)$. $P(C|w_t)$ adalah representasi sebuah *word* w_t memberikan kontribusi klassifikasi pada kelas C.

Ada perbedaaan mendasar antara *distributional clustering* dengan beberapa pendekatan *machine learning*, seperti k-nearest neighbor, adalah didalam menentukan *similarty metric* [Baker-1998]. *Similarty metric* yaitu ukuran kemiripan *feature* yang didasarkan pada variabel target yang akan ukur, bukan pada "atribut" masukan. Pada *distributional clustering* akan dilakukan evaluasi distribusi probabilitas *induced variable target* dengan *event* berbeda yang akan diklaster. Selanjutnya akan diukur kemiripan antara *event-event* seperti pada kemiripan distribusi *induced variable target*

3.2.3. Algoritma Divisive Information Theoretic Feature Clustering

Divisive Information Theoretic Feature Clustering dikembangkan oleh Inderjit S Dillon dkk [Dhill-2003]. Algoritma ini salah satu bentuk algoritma *feature clustering* yang menggunakan *hard clustering*, dimana setiap *word* hanya tergabung pada satu klaster saja. Algoritma ini adalah salah satu bentuk algoritma klastering yang divisive dimana jumlah klaster yang akan dibentuk didefinisikan pada awal mulai pembentukan klaster

$$W = \bigcup_{t=1}^{k} W_t \quad W_i \cap W_i = \phi, i \neq t \tag{3.5}$$

C adalah variable random diskrit yang nilainya berasal dari himpunan *class* (kategori/topik dokumen) $C = \{c_1, c_2, \ldots, c_l\}$ dan misalkan w_t adalah variable random yang nilainya berasal dari himpunan kata-kata (*term/feature*)

W = $\{w_1, w_2, \ldots, w_m\}$. Distribusi gabungan, $p(C, w_t)$ dapat diperoleh dari dokumen *training set*.

Algorithm Divisive_KL_Clustering(*P* ,ɳ,*l*,*k*,*W*)
Input: *P* adalah himpunan distribusi, $\{p(C \mid w_t) : 1 \le t \le m\}$,
ɳadalah himpunan *word prior*, $\{\pi_t = p(w_t) : 1 \le t \le m\}$,
l adalah jumlah kelas dokumen,
k adalah jumlah klaster yang diinginkan.
Output: *W* himpunan *word clusters* $\{W_1, W_2, \ldots, W_k\}$.

1. **Inisialisasi**: untuk setiap *word wt* , masukkan *wt* ke *Wj* sedemikian hingga $p(c_j \mid w_t) = \max_i p(c_i \mid w_t)$. proses ini akan memberikan *l* inisial *word clusters*; jika $k \ge l$ pecah setiap klaster menjadi paling sedikit $\lfloor k/l \rfloor$ klaster, sebaliknya gabungkan *l* klaster untuk mendapatkan *k word clusters*.
2. **Untuk setiap klaster *Wj*** , dihitung

$$\pi(W_j) = \sum_{w_t \in W_j} \pi_t \quad \text{dan} \quad p(C \mid W_j) = \sum_{w_t \in W_j} \frac{\pi_t}{\pi(W_j)} p(C \mid w_t)$$

3. **Hitung ulang semua klaster**: Untuk setiap *word wt* , indek klaster baru dihitung dengan cara:

$$j^*(w_t) = \arg\min_j KL(p(C \mid w_t), p(C \mid W_j))$$

Sehingga *word cluster* baru $W_j, 1 \le j \le k$, sebagai

$$W_j = \{w_t : j^*(w_t) = j\}.$$

4. **Iterasi dihentikan jika selisih fungsi objektif sekarang dengan fungsi objectif sebelumnya bernilai kecil** (misalnya 10^{-3});

$$of = \sum_{j=1}^{k} \sum_{w_t \in W_j} \pi_t KL(p(C \mid w_t), p(C \mid W_j))$$

jika sebaliknya kembali ke langkah 2.

Gambar 3.6 Algoritma Divisive Information Theoretic Feature Clustering

Divisive Information Theoretic Feature Clustering membentuk klaster kata-kata tersebut dalam sejumlah *k word cluster* $W = (W_1, W_2, \ldots, W_k)$. Algoritma ini menggunakan model *hard clustering* dimana nantinya setiap *word* w_t hanya terdapat pada satu *word cluster* (W).

Input yang dibutuhkan oleh algoritma ini adalah distribusi *conditional probability* setiap *word* terhadap masing-masing *class*, $p(C|w_t)$, bobot dari masing-masing *word* (*word prior probability*), $p(w_t)$, jumlah klaster yang akan

dibentuk serta kategori dokumen yang digunakan. Tahapan inisialisasi pada algoritma ini menjamin bahwa setiap $p(C|w_t)$ berada paling sedikit pada satu distribusi klaster $p(C|W_j)$, sehingga tidak akan berakibat pada nilai tak hingga perhitungan *KL-divergence* (langkah 3 dan 4) karena nilai $p(C|W_j)$ tidak pernah bernilai 0.

Langkah 1 pada algoritma diatas digunakan untuk membentuk *initial word cluster*. Jumlah *initial word cluster* sama dengan jumlah kategori (*class*). Sebuah *word* (w_t) termasuk dalam class C jika probabilitas kondisional $p(C| w_t)$ pada class C bernilai paling tinggi bila dibandingkan dengan class lainnya. Hal ini menjamin bahwa setiap *word* termasuk dalam minimal 1 klaster. Jika jumlah klaster yang akan dibentuk lebih dari jumlah class maka tiap-tiap *initial word cluster* dibagi menjadi jumlah klaster yang akan dibentuk, jika melebihi maka akan beberapa *initial word cluster* digabung.

Langkah 2 pada algoritma diatas digunakan untuk menghitung nilai *word cluster prior*. *Cluster Prior* merupakan *prior probabiliy* dari tiap *word cluster*. Nilai *Cluster Prior* diperoleh dari penjumlahan semua *prior probabiliy* setiap *word* ($p(w_t)$) yang berada pada klaster tersebut. *Cluster Prior* digunakan untuk menghitung probabilitas kondisional *word cluster* untuk tiap-tiap *class* $p(C|W_j)$.

Sebuah *word* w_t termasuk dalam *word cluster* tertentu dihitung dengan cara mengukur jarak antara w_t dengan setiap *word cluster* dalam koordinat ruang *term feature* multidimensi (*multidimensional feature space*). Sebuah *word* w_t termasuk dalam sebuah *word cluster* apabila jarak antara w_t dengan *word cluster* tersebut adalah jarak yang terdekat bila dibandingkan jarak antara w_t dengan *word*

cluster yang lainnya. Untuk mengukur jarak antara w_t dengan sebuah *word cluster* dihitung dengan menggunakan KL-*Divergence* (langkah 3).

Selama proses klastering akan ada penurunan *mutual information*. Idealnya dalam membentuk *word cluster* harus dapat dipertahankan laju penurunan *mutual information* sekecil mungkin. Kriteria global (fungsi objektif) digunakan untuk menentukan kualitas *word cluster* pada setiap iterasi. Fungsi objektif ini dapat diformulasikan dengan *jensen-shannon divergence* untuk setiap *word cluster*. Proses penurunan *mutual information* selama proses klastering diukur dengan menggunakan teorema 1 sebagai berikut:

Teorema 1 Perubahan mutual information selama proses klastering didefinisikan sebagai berikut:

$$I(C;W) - I(C;W^C) = \sum_{j=1}^{k} \pi(W_j) JS_{\pi}(\{p(C \mid w_t) : w_t \in W_j\}) \qquad (3.6)$$

dimana:

$\pi(W_j) = \sum_{w_t \in W_j} \pi_t$, $\pi_t = p(w_t)$, $\pi_t^{'} = \frac{\pi_t}{\pi(W_j)}$ untuk $w_t \in W_j$ dan JS adalah generalisasi *Jensen-Shannon Divergence* yang didefinisikan diatas

Ukuran global tentang kualitas word clusters yang ditunjukan pada gambar diatas dapat diinterpretasikan sebagai berikut:

- Kualitas dari sebuah word cluster (W_j) diukur dengan menggunakan Jensen Shannon divergence antara distribusi probabilitas masing-masing term (individual word distributions) $p(C|w_t)$ yang diberi bobot dengan word prior, $\pi_t = p(w_t)$

- Kualitas keseluruhan *word clustering* (fungsi objektif) diukur berdasarkan penjumlahan kualitas masing-masing *word clusters* (yang diberi bobot dengan *cluster prior*, $\pi(W_j) = p(W_j)$.

$$Q(\{W_j\}_{j=1}^{k}) = I(C;W) - I(C;W^C) \qquad (3.7)$$
$$= \sum_{j=1}^{k} \sum_{w \in W_j} \pi_t KL(p(C \mid w_t), p(C \mid W_j))$$

Nilai fungsi objektif selain dapat diukur dengan rumus seperti pada teorema di atas, juga dapat diukur menggunakan rumus KL-*Divergence*. Algoritma diatas secara iteratif akan mengulangi proses-proses (i) menggabungkan setiap *word* dengan *word cluster* berdasarkan kedekatan *word* dengan *word cluster* (yang diukur dengan KL-*Divergence*), dan (ii) membentuk *word cluster* baru serta menghitung distribusi setiap *word cluster* terhadap *class* $p(C|W_j)$.

Nilai fungsi objektif pada setiap iterasi (langkah 2-4). Proses iterasi akan dihentikan jika selisih nilai fungsi objektif iterasi yang terakhir bernilai kecil (pada tahap uji coba digunakan nilai 10^{-3} sama dengan yang direkomendasikan oleh Dhillon [Dhillon-2003]) dengan nilai fungsi objektif sebelumnya.

3.2.3.1 Klasifikasi Naive Bayes Menggunakan Word Cluster

Word cluster dapat digunakan sebagai pengganti *word* dalam algoritma *naïve bayes*. Hal ini dilakukan dengan cara melakukan estimasi parameter baru $p(W_s|c_i)$ untuk *word clusters*. Parameter ini digunakan untuk menggantikan probabilitas $p(w_t|c)$. $P(W_s|c_i)$ didefinisikan sebagai berikut [Dhill-2003]:

$$p(W_s \mid c_i) = \frac{\sum_{d_j \in c_i} n(W_s, d_j)}{\sum_{s=1}^{k} \sum_{d_j \in c_i} n(W_s, d_j)} \tag{3.11}$$

Sehingga klasifikasi dokumen *naïve bayes* dengan *word cluster* sebagai pengganti *word* [Dhill-2003] diformulasikan sebagai berikut:

$$c^*(d) = \arg\max ci\left(\frac{\log p(ci)}{|d|} + \sum_{s=1}^{k} p(Ws \mid d) \log p(Ws \mid ci)\right) \tag{3.12}$$

dimana $p(Ws \mid d) = \frac{n(Ws \mid d)}{|d|}$

3.3. HIBERNATE

Hibernate adalah salah satu teknologi *object relational persistence mapping* (ORM) dan *query service*. Transformasi persistent data dari sebuah *representasi* data (*relational database*) ke sebuah *representasi* data yang lain (*java objects*) ataupun sebaliknya adalah fungsi yang sebenarnya dilakukan oleh ORM.

Dalam aplikasi pemrograman yang berorientasi objek, *persistence* memperbolehkan sebuah objek untuk bisa tetap ada diluar proses aplikasi yang membuatnya [King-2005]. *State* dari objek tersebut disimpan pada *datastore* dan objek dengan *state* yang sama suatu saat dapat di buat kembali jika aplikasi dijalankan lagi.

Transient Object adalah object yang dibuat dimana *state* dari objek ini tidak disimpan pada *datastore*, sehingga tidak dapat dibuat kembali sebuah objek dengan *state* yang sama suatu saat. Masa hidup sebuah *transient object* terbatas

pada proses yang membuatnya. Kebanyakan objek pada aplikasi terdiri dari gabungan *persistent* dan *transient* objek sehingga diperlukan sebuah sub-sistem yang menangani *persistent* objek.

Domain model adalah representasi dari *business entities* yang digunakan dalam aplikasi java. Dalam arsitektur aplikasi yang terbagi menjadi beberapa layer, *domain model* digunakan untuk menjalankan *business logic* pada *business layer* (pada java bukan pada database). *Business layer* berhubungan dengan *persistence layer* untuk proses *loading* dan menyimpan *persistent object* dari *domain model.*

ORM adalah *middleware* pada *persistence layer* yang menangani *persistent object.* ORM bukan merupakan satu-satunya solusi permasalahan yang digunakan untuk menangani *persistent object*, cara lain yang bisa digunakan adalah dengan EJB atau langsung memakai SQL yang mengkopi nilai yang dari JDBC result set pada *persistent object* di *domain model.*

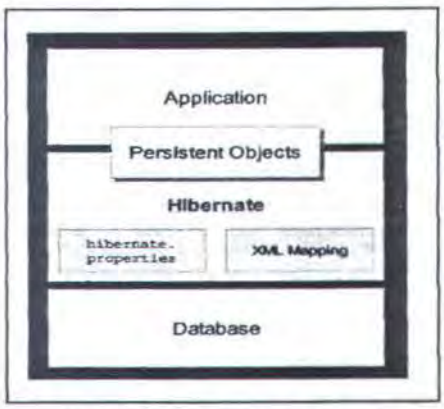
Hibernate ditulis dengan menggunakan bahasa pemrograman java. *Persistent object* dapat dibuat menggunakan hibernate dengan idiom bahasa pemrograman java termasuk didalamnya *association, inheritance, polymorphism, composition, java collection framework* seperti Set, List, Map. Hibernate adalah *free software* yang menggunakan lisensi LGPL.

Hibernate mendukung banyak DBMS (*Database Management System*) antara lain Oracle, DB2, MySQL, PostgreSQL, Sybase, SAP DB, HypersonicSQL, Microsoft SQL Server, Posgress, McKoi SQL, Pointbase dan Interbase.

File konfigurasi dengan format XML digunakan untuk memetakan (*mapping*) antara objek java dengan data model pada *relational database*, selain *object oriented query language* yang mirip dengan *SQL-Query language* juga digunakan oleh hibernate. Hibernate mengambil alih fungsi mapping dari java class ke table dalam database dan demikian juga sebaliknya dari tipe data java ke tipe data SQL.

Object Oriented Query Language (OOQL) yang digunakan oleh hibernate mendefinisikan *join table* sebagai *object property path*. Selain itu OOQL mendukung fungsi dan operator SQL, fungsi aggregat seperti *sum, avg, min, max, count*. OOQL mendukung fungsi *group by, having* dan *order by*. Hibernate juga mendukung *sub query* (*sub select* pada database). *Query* yang dihasilkan oleh hibernate mengembalikan nilai berupa Objek atau nilai skalar. Hibernate yang digunakan dalam tugas akhir ini adalah versi 1.25 dan versi terbaru dari hibernate pada saat tugas akhir ini ditulis adalah versi 3.0.

3.3.1 Arsitektur Hibernate

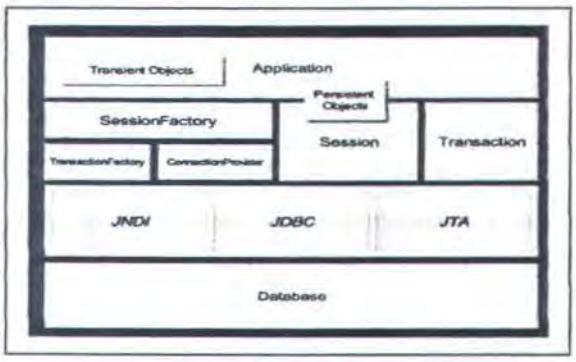


Gambar 3.7. Arsitektur Hibernate

Arsitektur tingkat tinggi dari hibernate didefinisikan seperti gambar 3.7. Pada gambar tersebut ditunjukkan bahwa hibernate menggunakan file konfigurasi (hibernate.properties dan *xml mapping*) untuk menyediakan layanan persisten dan objek persistent dari dan kepada aplikasi.

Sekumpulan *Application Programming Interface* (API) yang digunakan oleh *business layer* yaitu:

- Interface yang digunakan untuk melakukan operasi CRUD dan Query. Interface ini merupakan main point antara business logic dengan Hibernate. Interface tersebut adalah `Session`, `Transaction` dan `Query`.
- Interface yang digunakan oleh aplikasi untuk melakukan konfigurasi Hibernate, yang paling penting adalah class `Configuration`.
- Callback Interface yang digunakan oleh aplikasi untuk event yang yang terjadi didalam Hibernate seperti `Interceptor`, `Lifecycle`, dan `Validatable`.
- Interface yang merupakan *implementasi* mapping *Hibernate* seperti `UserType`, `CompositeUserType`, dan `IdentifierGenerator`.



Gambar 3. 8. Arsitektur Keseluruhan Hibernate

Definisi dari *Interface* yang terdapat pada gambar diatas adalah sebagai berikut:

- `SessionFactory`, factory untuk `Session`, client dari `ConnectionProvider`.
- `Session`, class yang digunakan antara aplikasi dengan database. `Session` mengelola object untuk aplikasi. `Session` adalah *wrapper* dari sebuah koneksi JDBC.
- Persistent Object & Collections, object yang memiliki *persistent state* dan bisnis proses, bisa berupa JavaBean. *Persistent Object & Collection* selalu mempunyai 1 Session.
- Transient Object & Collections, merupakan instance dari *persistent class* yang tidak terkait dengan Session. Transient Object & Collections bisa berupa object yang di buat oleh aplikasi yang di-*persistent*-kan. Atau obyek yang session telah di tutup (*closed session*).
- `Transaction`, object yang digunakan oleh aplikasi untuk menspesifikasikan unit kerja yang atomic. `Transaction` adalah abstraksi dari JDBC, JTA atau CORBA *Transaction*. Sebuah `Session` dapat memiliki lebih dari 1 `Transaction`.
- `ConnectionProvider`, yaitu abstraksi dari `DataSource` atau `DriverManager`. `ConnectionProvider` tidak di bisa diakses langsung oleh *business layer*.
- `TransactionFactory`, factory untuk `Transaction`.

Sebuah instance dari `Datastore` adalah representasi dari keseluruhan *mapping* antara objek-objek java ke database relasional. *Mapping* di *compile* dari

berbagai file *mapping* yang berformat xml. Sebuah instance dari `Datastore` dapat dibuat dengan memanggil fungsi `Hibernate.createDatastore()`.

```
Datastore datastore =
      Hibernate.createDatastore().storeInputStream(inputStream);

// atau

Datastore datastore =
      Hibernate.createDatastore().storeFile("ta.hbm.xml");
// inisialisasi SessionFactory
SessionFactory sessionFactory =
datastore.buildSessionFactory();
```

Gambar 3. 9. Inisialisasi Datastore.

Pada gambar 3.19 ditunjukkan cara inisialisasi untuk mendapatkan Datastore. `SessionFactory` dapat diperoleh segera setelah instance dari `Datastore` selesai di buat. Sedangkan cara untuk memperoleh SessionFactory ditunjukkan pada baris code setelah Datastore dibuat.

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-1.1.dtd">
<hibernate-mapping>
      <!-- category -->
      <class name="ta.base.persistent.model.CClass" table="CCLASS">
            <id name="cclassCode" column="CCLASS_CODE" type="long">
                  <generator class="cirrus.hibernate.id.HiLoGenerator"/>
            </id>
            <property name="cclassName" column="CCLASS_NAME"
                  type="string"/>
            <set role="documents" table="DOCUMENT" lazy="true" order-
                  by="DOCUMENT_NAME">
                  <key column="CCLASS_CODE"/>
                  <one-to-many  class="ta.base.persistent.model.Document"/>
            </set>
      </class>
<hibernate-mapping>
```

Gambar 3.10. XML *mapping java-relational database* yang digunakan oleh Hibernate

File xml digunakan untuk melakukan konfigurasi *object relational mapping*. File ini kemudian digunakan sebagai komunikasi antara objek java

dengan data model pada database. Selain itu dengan bantuan tool yang ada dimiliki hibernate file *mapping* dapat digunakan untuk membuat *java class* dan table database. Contoh file xml ditunjukkan pada gambar 3.10.

Sebuah *object* (*entity instance*) dapat berupa *transient* atau *persistent* tergantung pada `Session`. Instance obyek yang dibuat merupakan *transient* obyek. `Session` dapat merubah sebuah *transient* objek menjadi *persistent* objek. `Session` memiliki method `load()` yang digunakan untuk mengambil sebuah *persistent instance* jika *indentifier*nya diketahui. Berikut ini cara untuk dilakukan untuk membuat *persistence persistent object*:

```
CClassModel classModel = new CClassModel();
classModel.setCclassName(className);
classModel.setDocumentModels(documents);
Long generatedid = (Long) sess.save(classModel);
```

Gambar 3.11. Merubah transient object menjadi persistent object

```
CClassModel model = (CClassModel) sess.load(CClassModel.class,
generatedid );
// wrap primitive keys
long classModelCode = 1;
CClassModel model = (CClassModel) sess.load(CClassModel.class,
new Long(classModelCode) );
```

Gambar 3.12. *Loading persistent object* dari database

`Session.delete()` digunakan untuk menghapus obyek dari database. Selain itu banyak obyek dapat dihapus dengan menggunakan *Hibernate Query Language* ke `Session.delete()`.

```
sess.delete(cat);
```

Gambar 3.13 Menghapus *peristent object* dengan Session.delete()

Object Oriented Query Language (OOQL) disediakan oleh hibernate untuk digunakan melakukan query. Method `find()` pada `Session` bisa dilakukan untuk melakukan query. Query yang dilakukan menghasilkan sebuah obyek atau sekumpulan objek yang berupa List. Method `iterate()` bisa digunakan jika query yang dijalankan menghasilkan list obyek yang besar yang tidak semuanya digunakan. Method `iterate()` yang mengembalikan objek berupa `java.util.Iterator`. Berikut ini contoh query dengan *Hibernate Query Language*.

```
String sql =
" from obj in class ta.base.persistent.model.DocumentModel " +
" where obj.cclassModel.cclassCode = ? ";
List documentModels = session.find(sql, new Long(cclassModelCode),
Hibernate.LONG);
Iterator i = session.iterate(sql, new Long(cclassModelCode),
Hibernate.LONG);
while(i.hasNext()){
      DocumentModel model = (DocumentModel) i.next();
      // do additional business process here
}
```

Gambar 3.14 Contoh pengunaan Hibernate Query Language

Sebuah objek tetap berupa *persistent object* selama `Session` yang meng-*instantiate*nya belum ditutup. `Session.flush()` perlu dijalankan untuk menyakinkan bahwa segala perubahan disinkronisasi dengan database. Jika digunakan `Transaction` API, maka `Session.flush()` tidak perlu dilakukan. Sebagai penggantinya digunakan `Transaction.commit()` yang terdapat pada `Transaction`. *Flushing* `Session` dan *commit* akan dijalankan jika method `Transaction.commit()` dipanggil. Jika *Transaction* API tidak digunakan maka

bisa pakai `Session.connection().commit()` untuk melakukan *commit* ke database

`Transaction.rollback()` atau `Session.connection().rollback()` digunakan untuk membatalkan *transaction* yang belum di-*commit*. Untuk menutup `Session` dilakukan dengan memakai *method* `Session.close()`.

```
Session sess = factory.openSession();
try {
    //do some work
    ...
    sess.flush();
    sess.connection().commit();
}catch (Exception e) {
    sess.connection().rollback();
    throw e;
}finally {
    sess.close();
}
```

Gambar 3.15 Tahapan menggunakan Session

Sebuah file konfigurasi database digunakan untuk konfigurasi `Datastore` selain file xml yang digunakan sebagai *mapping* antara tabel dalam database dengan class. File konfigurasi `Datastore` diperlukan jika *connection* ke database tidak dilakukan secara manual oleh *user*. Contoh file konfigurasi dapat dilihat pada gambar 3.27. File konfigurasi untuk `SessionFactory` dapat dilakukan dengan berbagai cara antara lain:

- Parameter yang berupa instance dari `java.util.Properties` diparsing ke `Datastore.buildSessionFactory()`.
- File hibernate.properties diletakan pada root dari classpath.
- Setting system properties dengan perintah "`java -Dproperty=value`".
- Setup konfigurasi pada tag <property> di dalam file `hibernate.cfg.xml`.

```
PropertiesLoader loader =
        PropertiesLoader.instance();
Properties properties = Environment.getProperties();
String hbmFile =
  properties.getProperty("hbmFile",    "ta.hbm.xml");
ClassLoader classLoader = Persistent.class.getClassLoader();
InputStream inputStream =
classLoader.getResourceAsStream(hbmFile);
// create data store
datastore =
      Hibernate.createDatastore().storeInputStream(inputStream);
// build session factory
sessionFactory = datastore.buildSessionFactory();
```

Gambar 3.16 Inisialisasi SessionFactory

Tabel dibawah adalah beberapa parameter yang nilainya perlu diset pada file konfigurasi yaitu:

Tabel 3.12 Parameter setting Hibernate

Property	Keterangan
hibernate.connection.driver_class	*jdbc driver class*
hibernate.connection.url	*jdbc url*
hibernate.connection.username	*database user*
hibernate.connection.password	*user password*
hibernate.connection.pool_size	Jumlah maksimum Connection dalam *pool*
hibernate.statement_cache.size	Jumlah maximum cached PreparedStatements (harus bernilai 0 untuk Interbase)
hibernate.connection.isolation	*transaction isolation level (optional)*
hibernate.connection.xxxx	*pass the JDBC property xxxx to DriverManager.getConnection()*

hibernate.dialect	dialek sql yang digunakan oleh database (setiap database memiliki platform sql yang spesifik)

```
################ Pooled Conn#############
hibernate.dialect=cirrus.hibernate.sql.MySQLDialect
hibernate.use_outer_join=true
hibernate.connection.driver_class=com.mysql.jdbc.Driver
hibernate.connection.url=jdbc:mysql://127.0.0.1/tugasakhir
hibernate.connection.username=heru
hibernate.connection.password=heru
hibernate.connection.pool_size=60
hibernate.statement_cache.size=80
################################################
```

Gambar 3.17 konfigurasi Hibernate dengan file hibernate.properties

Sedangkan gambar 3.17 adalah contoh file hibernate.properties yang digunakan dalam tugas akhir ini.

BAB IV
PERANCANGAN PERANGKAT LUNAK

Pada bab ini dibahas mengenai perancangan perangkat lunak yang digunakan untuk mengimplementasikan algoritma *divisive feature clustering*. Konsep perancangan yang digunakan adalah konsep *Visual Modeling*, yaitu metode atau cara berpikir tentang suatu masalah dengan menggunakan pemodelan. Model berguna untuk memahami kebutuhan, komunikasi, dokumentasi desain program dan database. Konsep *Visual Modeling* diterapkan dengan menggunakan *Unified Modeling Language* (UML). Notasi-notasi mulai dari fase analisa, desain dan implementasi disediakan oleh UML. Bab ini dibagi menjadi tiga tahap/bagian, yaitu tahap definisi sistem, *use case view* dan *logical view*.

4.1. DEFINISI SISTEM

Sistem yang akan dibuat adalah aplikasi klasifikasi dokumen dengan menggunakan algoritma *divisive information theoretic feature clustering* sebagai pembentuk *word cluster*. *Word cluster* yang selanjutnya digunakan sebagai pengganti parameter *term* dalam klasifikasi dokumen. Untuk algoritma klasifikasi dokumen digunakan algoritma klasifikasi *naïve bayes*.

Ada 5 elemen besar yang berperan dalam sistem yaitu:

- Indexing. Bagian dari perangkat lunak yang digunakan untuk melakukan indek dataset. Hasil keluaran dari *indexing* berupa *term-term* yang digunakan dalam proses clustering.

- Clustering. Bagian dari perangkat lunak yang digunakan untuk membentuk klaster. Hasil keluaran dari *clustering* dipakai oleh algoritma klasifikasi.
- Classification. Digunakan untuk melakukan klasifikasi dokumen.
- Persistent. Digunakan untuk penyimpanan dan pengambilan data dari database
- Antar muka. Berfungsi sebagai menampung input dan menampilkan output . Merupakan tampilan yang berhubungan langsung dengan pengguna..

4.2. PEMBUATAN USE CASES VIEW

Tahap ini digunakan untuk mendefinisikan tingkah laku sistem dengan fungsi-fungsi yang disediakan olehnya. Proses-proses yang dilakukan dalam mendefinisikan sistem adalah dengan melakukan identifikasi actor yang terlibat, use case untuk setiap actor dan activity diagram untuk use case yang memerlukan penjelasan lebih rinci.

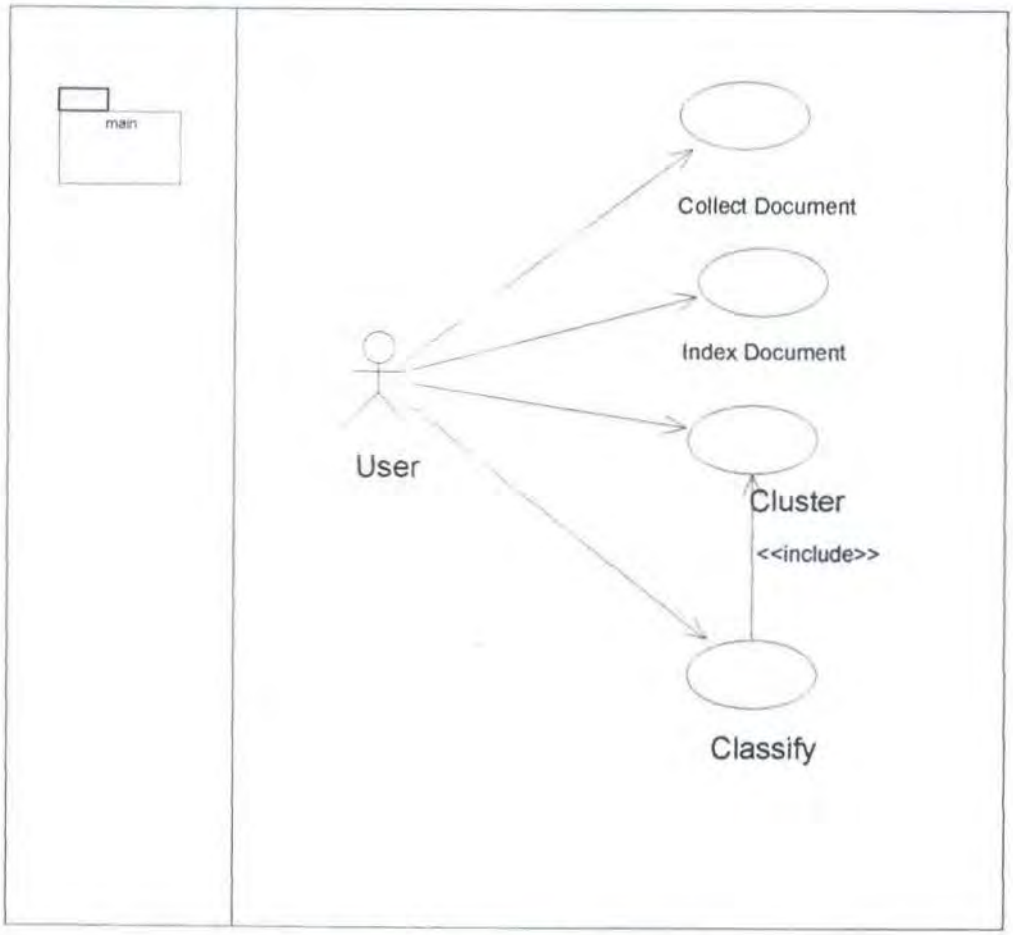
4.2.1. Actor

Actor dalam terminologi UML adalah segala sesuatu yang berada diluar sistem yang berinteraksi dengan sistem [Richt-1999]. Dengan memodelkan *actor* dapat diketahui dengan jelas segala sesuatu yang berinteraksi dengan sistem dan bagaimana cara berinteraksinya. Pada sistem ini terdapat seorang *actor* yang berhubungan dengan sistem.

4.2.2. Use Cases

Interaksi-interaksi yang dilakukan *actor* ke sistem direpresentasikan dalam *use case* [Richt-1999]. Setiap interaksi yang dilakukan *actor* adalah *use case* atau

bagian dari sebuah *use case*. Gambar berikut ini adalah *use case* dari sistem yang akan dibuat.



Gambar 4.1. Diagram *use case* sistem

- Use Case Collect Document

Proses pengumpulan dokumen yang akan digunakan sebagai dataset *training* dilakukan *Indexing Manager*. Yang termasuk dalam proses pengumpulan dokumen antara lain proses pencatatan penyimpanan data-data dokumen yang disimpan dalam *hard drive* komputer.

- Use Case Index Document

Proses pengindeksan dataset *training* digunakan untuk mendapatkan *term-term*. Kumpulan *terms* ini digunakan sebagai data masukan untuk membentuk klaster.

- Use Case cluster

Use cluster digunakan untuk membentuk klaster dari kumpulan *term-term*.

- Use Case classify

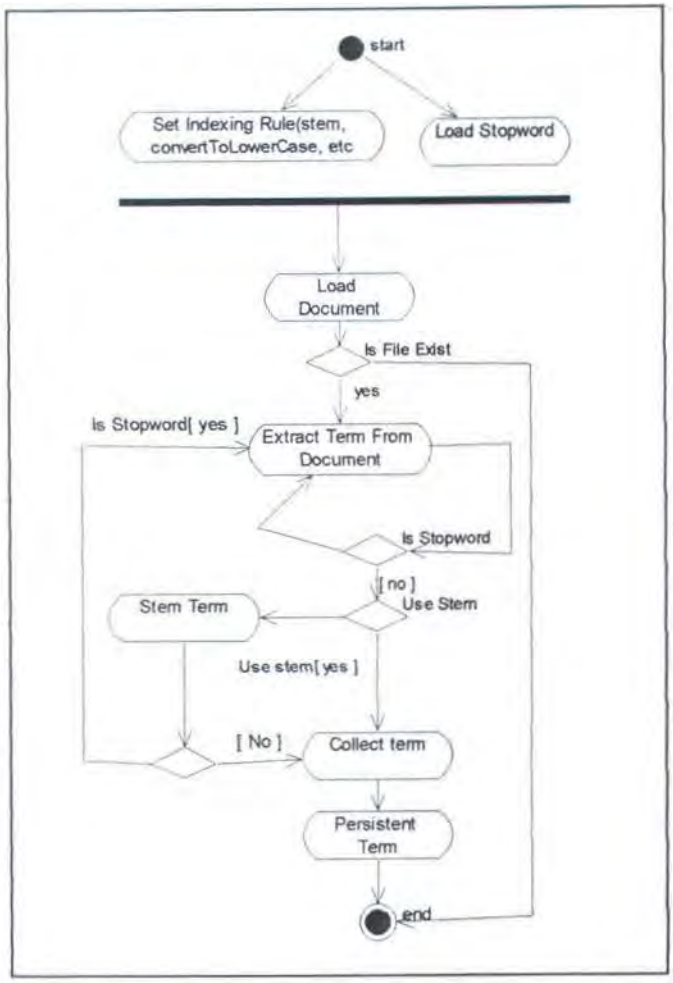
Use Case classify digunakan untuk melakukan klasifikasi terhadap dokumen yang belum terlabeli. *Use case classify* ini *include* dengan *use case cluster* karena Klaster-klaster yang dihasilkan oleh *use clase cluster* akan digunakan oleh *use case classify* sebagai data untuk menentukan kategori dari sebuah dokumen

4.2.3. Activity Diagram

Aliran kerja (*wokflow*) aktifitas dari sebuah *use case* dijelaskan dalam [Richt-1999] *Activity Diagram*. Definisi urutan aktifitas yang harus dilakukan dalam sebuah *use case* digambarkan dalam *Activity Diagram*. Pada bagian ini akan dijelaskan beberapa *Activity Diagram* dalam *use case*.

4.2.3.1. Index Document

Pada gambar 4.2 ditunjukkan diagram alur kerja proses pengindeksan dokumen. Proses ini diawali dengan melakukan setting rule pengindeksan dokumen (menggunakan stemming, menghilangkan header (jika menggunakan dataset 20NG), konversi huruf, melakukan pruning dan lain sebagainya) dan melakukan *loading* stopword.



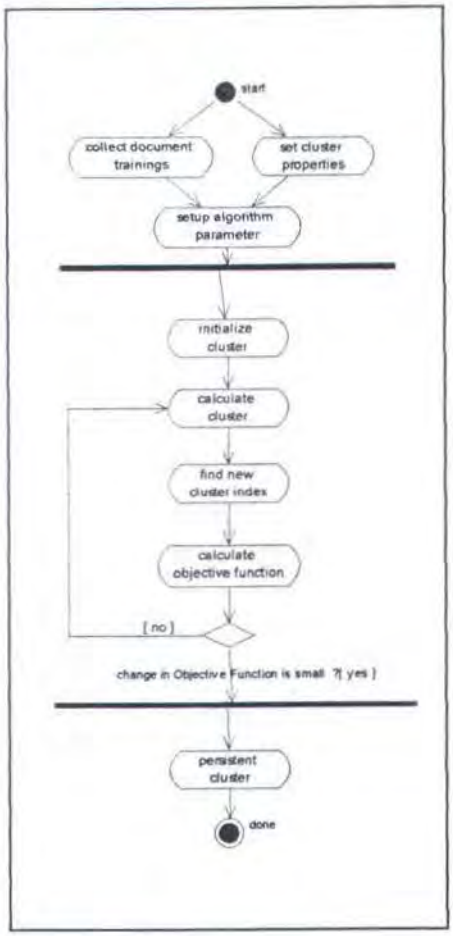
Gambar 4.2. Activity Diagram untuk Indexing Document

Load document digunakan untuk melakukan *loading* dokumen yang akan diindek serta melakukan ekstraksi dari setiap *term* yang ada didokumen tersebut. Setiap *term* yang diperoleh di konversi ke huruf kecil atau huruf besar dan jika digunakan *stemming* maka *term* tersebut di-*stem*. Jika *term* tersebut bukan *stopword* maka *term* tersebut merupakan *term* dokumen.

4.2.3.2. Clustering

Aktivitas ini diawali dengan proses *set cluster properties,* yaitu memberikan nilai dari beberapa parameter yang diperlukan oleh algoritma

Divisive Information Theoretic Feature Clustering yaitu kategori dokumen yang digunakan dalam membentuk klaster, jumlah klaster yang ingin dibentuk.



Gambar 4.3. Activity Diagram untuk Cluster Term

Aktivitas yang dilakukan bersamaan dengan proses *set cluster properties* adalah proses *collect document* training, dengan memilih beberapa dokumen dari tiap kategori yang digunakan sebagai dataset. Dokumen-dokumen ini diambil dari lebih dari 1 kategori dokumen, dimana kategori-kategori ini sudah ditentukan sebelumnya.

Proses selanjutnya adalah *setup algorithm parameter*, aktivitas ini terkait dengan beberapa proses untuk membentuk parameter-parameter yang diperlukan

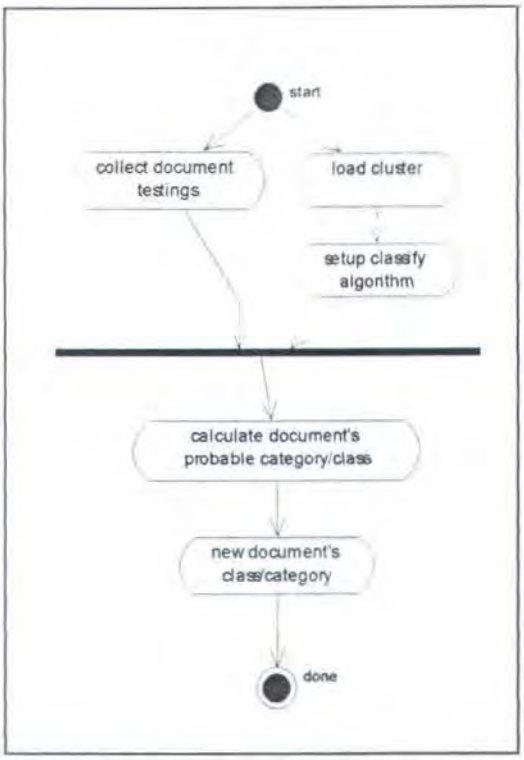
dalam algoritma ini yaitu proses perhitungan *conditional probability* setiap *term* pada tiap-tiap kategori (*class)* dokumen, perhitungan *prior probability* setiap *term*.

Ketiga aktivitas yang telah disebutkan diatas, diteruskan oleh proses pembentukan klaster-klaster. Proses ini dilakukan oleh beberapa aktivitas mulai dari *initialize cluster* yang dilanjutkan dengan *calculate cluster* hingga aktivitas perhitungan fungsi objektif (*calculate objective function*). Aktvitas *calculate cluster* sampai dengan aktivitas *calculate objective function* akan terus dilakukan selama nilai selisih fungsi objektif iterasi ke-*i* dengan nilai fungsi objektif iterasi ke-(*i*-1) lebih besar dari batasan parameter yang diberikan

4.2.3.3. Classify

Aktivitas *classify* dilakukan setelah pembentukan klaster-klaster yang dilakukan oleh aktivitas *clustering* selesai. Proses ini diawali dengan mengumpulkan dokumen yang akan diuji (*collect document testing*). Selain itu juga dilakukan proses *load cluster*, yang akan digunakan untuk melakukan klasifikasi. Proses ini dilanjutkan dengan proses setting nilai parameter yang digunakan algoritma klasifikasi

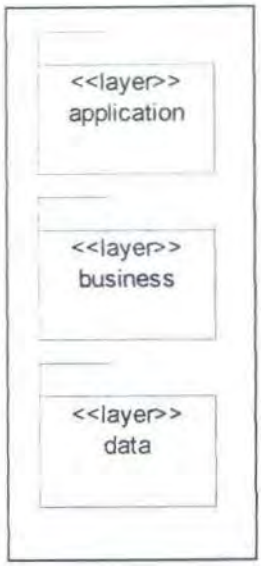
Selanjutnya dilakukan proses klasifikasi dengan menggunakan klaster. Hasil perhitungan yang diperoleh digunakan untuk menentukan kategori dari dokumen tersebut.



Gambar 4.4. Activity Diagram untuk Classify

4.3. PEMBUATAN LOGICAL VIEW

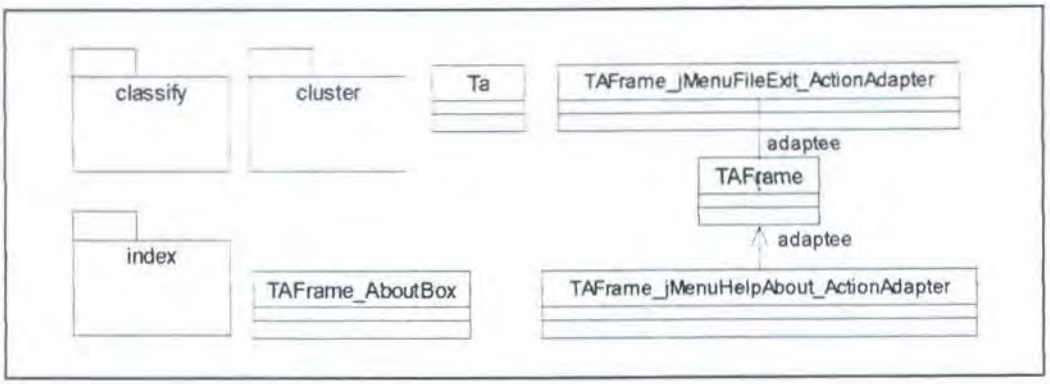
Pada tahapan ini dilakukan pembuatan *logical view*, yang berisi realisasi dari *use case* dalam *use case realization* dan membuat *sequence diagram* untuk menggambarkan *method-method* yang diperlukan oleh class-class yang akan menyusun aplikasi ini.



Gambar 4.5. Diagram *package* untuk struktur *Three Tier*

Class-class yang dibuat dikelompokan dalam *three-tier layer* menjadi *application layer*, *bussiness layer* dan *data layer*. Seperti terlihat pada gambar 4.5.

4.3.1. Application Layer



Gambar 4.6 Class Diagram untuk Application Layer

Application layer secara umum berfungsi sebagai *layer* interaksi antara pengguna dengan aplikasi. Form-form disediakan pada layer ini untuk digunakan oleh user. Data-data masukan yang diberikan oleh user diterima oleh *layer* ini untuk selanjutnya diproses oleh *business layer*. Data hasil pemrosesan yang dilakukan oleh *business layer* ditampilkan pada *layer* ini.

Application layer terdiri dari beberapa class yang dibagi menjadi beberapa *package*. Pembagian ini didasarkan pada pembagian fungsi yang ada *aplication layer*. Class diagram dari *application layer* bisa dilihat pada gambar 4.6.

4.3.1.1. Package ui

Package ini terdiri dari tiga *package* dan tiga class utama. Penjelasan mengenai ketiga *package* ini masing-masing ada dibagian lain dari bab ini. Berikut ini merupakan penjelasan dari masing-masing class yang ada dalam *package* ui. Class diagram dari *package* ui bisa dilihat pada gambar 4.6.

- class `Ta`

Class Ta adalah class utama yang digunakan untuk menjalankan aplikasi ini. Aplikasi dijalankan melalui class ini dengan membuat *instance* dari class `TAFrame`. Didalam class ini terdapat *method* main() yang digunakan sebagai *main* program.

- Class `TAFrame_AboutBox`

Class ini digunakan untuk menampilkan deskripsi tentang aplikasi ini. Class ini berupa turunan dari class `javax.swing.JDialog`. Deskripsi singkat dari class ini dapat dilihat pada class ini.

- Class `TAFrame_jMenuHelpAbout_ActionAdapter`

Class ini adalah *inner class* dari class `TAFrame`. Class ini digunakan sebagai *adapter class* untuk menu helpAbout.

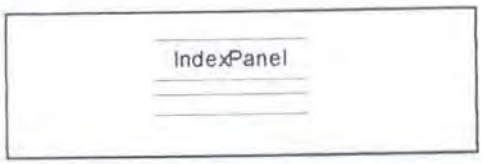
- Class `TAFrame_jMenuFileExit_ActionAdapter`

Class ini adalah *inner class* dari class `TAFrame`. Class ini digunakan sebagai *adapter class* untuk menu exit.

- Class TAFrame

Class ini adalah class utama layer aplikasi. Interaksi antara *user* dengan aplikasi dilakukan melalui class ini. *Proses-proses* pengindeksan dokumen, pembentukan cluster, dan klasfikasi juga dilakukan *user* pada class ini.

4.3.1.2. Package index



Gambar 4.7 Class Diagram package ta.ui.index

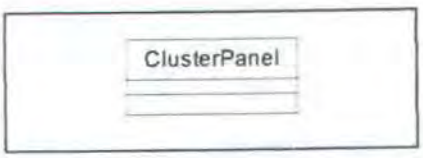
Package ini terdiri dari sebuah class yaitu class IndexPanel. Penjelasan menggenai class IndexPanel adalah sebagai berikut:

- Class IndexPanel

Class ini adalah class yang digunakan untuk menampilkan hasil pengindeksan dokumen yang dilakukan oleh *business layer*. Selain itu class ini juga digunakan *user* untuk melakukan proses pengindeksan dokumen.

4.3.1.3. Package cluster

Sama halnya *package* index, *package* ini juga terdiri dari sebuah class yaitu class ClusterPanel. Class diagram untuk *package* ini bisa dilihat pada gambar 4.8.



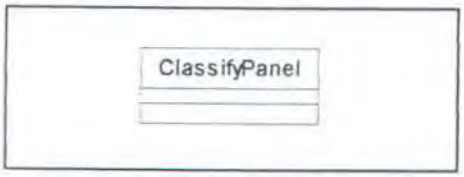
Gambar 4.8 Class Diagram package ta.ui.cluster

- Class ClusterPanel

Class ini digunakan untuk melakukan pembentukan *word cluster*. Hasil pembentukan *word cluster* ditampilkan pada class ClusterPanel.

4.3.1.4. Package classify

Package ini terdiri dari sebuah class yaitu class ClassifyPanel. Class diagram untuk *package* ini bisa dilihat pada gambar 4.9.



Gambar 4.9 Class Diagram package ta.ui.classify

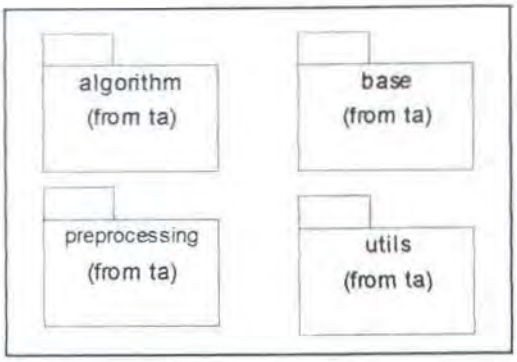
- Class ClassifyPanel

Class ini digunakan untuk melakukan klasifikasi menggunakan algoritma *naïve bayes* dengan *word cluster* sebagai parameternya. Dari panel ini ditampilkan hasil klasifikasi.

4.3.2. Bussiness Layer

Business layer secara umum berfungsi sebagai *layer* pengolahan data. Data yang ada di *data layer* diproses oleh *business layer* ditampilkan pada *application layer*. Demikian juga sebaliknya data masukan yang berasal dari *application layer* diproses oleh *bussiness layer* untuk selanjutnya disimpan di *data layer*.

Business layer terbagi menjadi beberapa class-class yang dikelompokkan menjadi beberapa package menurut fungsi masing-masing.



Gambar 4.10 Class diagram untuk bussiness layer.

4.3.2.1. Package Base

Base package terdiri dari 4 *interface*. *Persistent package* adalah *package* untuk data layer. Penjelasan mengenai *persistent package* ada pada sub bab data layer yang berada pada bagian lain dari bab ini. *Base package* dimaksudkan sebagai *class* atau *interface* dasar yang digunakan aplikasi.

Penggunaan beberapa interface dalam package ini selain dimaksudkan untuk menyembunyikan detail implementasi dari masing-masing *method* interface kepada *class* yang menggunakannya juga untuk merepresentasikan terminologi yang digunakan dalam *Information Retrieval*. Penjelasan masing-masing interface adalah sebagai berikut

- Interface `Feature`

Interface `Feature` adalah interface untuk *term* dalam terminologi *Information Retrieval*. Interface ini mempunyai *method* seperti `getFrequency()` untuk mendapatkan frekuensi *term*, `getFeature()` untuk mendapatkan nama *term*.

- Interface `Document`

Merupakan interface untuk dokumen. Setiap dokumen memiliki beberapa *feature* yang bisa didapatkan digunakan *method* `getFeatures()`. Selain itu dapat diperoleh frekuensi kemunculan sebuah *term* pada dokumen dengan digunakan *method* `getFeatureFrequency()`, method `getDocumentTitle()` digunakan untuk memperoleh nama dokumen tersebut, method `getDocumentClassCode()` digunakan untuk mendapatkan kategori dari dokumen tersebut.

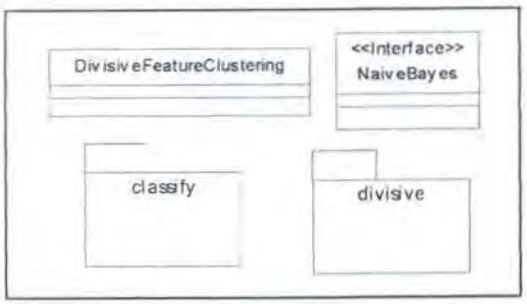
- Interface `Cluster`

Interface Cluster digunakan untuk merepresentasikan Cluster yang digunakan dalam algoritma divisive information theoretic feature clustering. Interface ini mempunyai *method* `calculateClusterPrior()` yang digunakan untuk menghitung *cluster prior* ($\pi(W_j) = \sum_{w \in W_j} \pi$), sedangkan *method* `calculateClassConditionalProbability()` untuk menghitung $p(C|W_j)$, serta beberapa method yang lainnya.

- Interface `CClass`

Digunakan untuk merepresentasikan Kategori dokumen (*document label*). Kategori dokumen adalah topik utama yang dibahas pada dokumen tersebut. Method `getClassFeatureConditionalProbability()` yang digunakan untuk mendapatkan nilai dari $p(C|w_t)$, sedangkan method `getFeatureFrequency()` yang digunakan untuk mendapatkan jumlah kemunculan *term* dalam suatu Kategori (jumlah *term* ini didapatkan dari *training dataset*.

4.3.2.2. Package algortihm



Gambar 4.11 Class Diagram untuk package algorithm

Package algorithm terdiri dari sebuah class, sebuah interface dan dua buah *package*. Penjelasan masing-masing adalah sebagai berikut

- Class `DivisiveFeatureClustering`

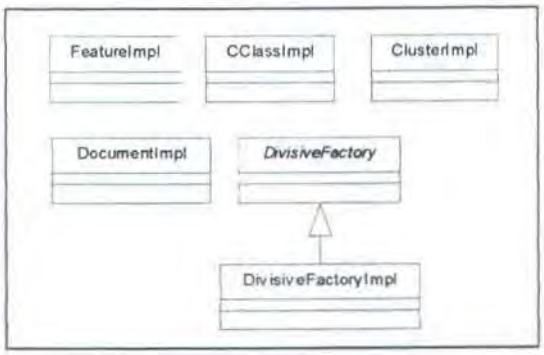
Class ini adalah implementasi dari algoritma *Divisive Information-Theoretic Feature Clustering*. Class ini berfungsi membentuk klaster-klaster *term-term* (Interface `Feature`) yang berasal masing-masing kategori (yang diimplementasikan dengan Interface `CClass`). *Class* ini memiliki *method* `getObjectiveFunction()` yang digunakan untuk mendapatkan nilai fungsi objektif, serta *method* `run()` yang digunakan untuk membentuk *word cluster*. Keseluruhan perhitungan dilakukan pada *method* `run()`, objek `Cluster` yang merupakan hasil dari *word cluster* yang telah dibentuk dikembalikan oleh method ini.

- Interface `NaiveBayes`

Interface ini adalah interface untuk algoritma klasifikasi *Naïve Bayes*. *Class* `NaiveBayesCluster` adalah implementasi dari *Interface* `NaiveBayes` untuk klasifikasi dengan menggunakan *word cluster*. *Interface* ini memiliki

method `classify()` yang digunakan untuk mengklasifikasi sebuah dokumen, *method* `getClassProbs()` yang digunakan untuk mendapatkan nilai probabilitas sebuah dokumen dalam kategori-kategori yang diberikan.

4.3.2.3. Package Divisive



Gambar 4.12 class diagram package divisive

Package Divisive adalah kumpulan class-class implementasi dari *Interface-interface* yang terdapat pada *package ta.base* untuk algoritma *Divisive Information Theoretic Feature Clustering*. *Package* ini terdiri dari lima Class yang penjelasannya masing-masing adalah sebagai berikut:

- Class `FeatureImpl`

  Class ini adalah Class *default* implementasi Interface `Feature`.

- Class `CClassImpl`

  Class ini adalah class implementasi Interface `CClass` yang berada pada *package* `ta.base.CClass`.

- Class `ClusterImpl`

  Merupakan implementasi dari *Interface* `Cluster`.

- Class `DocumentImpl`

  Class ini adalah implementasi *Interface* `Document`.

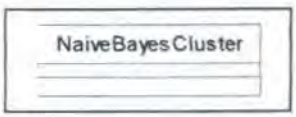
- Class `DivisiveFactory`

Abstract class yang berfungsi sebagai *factory method*. Class ini digunakan untuk membuat *concrete object* dari *Interface* `Feature`, `CClass`, `Cluster` dan `Document`. Dari Class ini dapat dibuat object yang merupakan instance dari `CClass` dengan *method* `createCClass()`, *membuat object* dari `Cluster` dengan *method* `createCluster()`, membuat *object* dari `Document` dengan *method* `createDocument()` dan membentuk *word cluster*.

- Class `DivisiveFactoryImpl`

Class ini merupakan *concrete* class dari Class `DivisiveFactory`

4.3.2.4. Package Classify

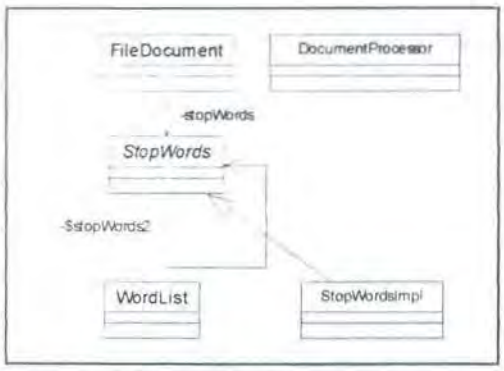


Gambar 4.13 class diagram package ta.algorithm.classify

Package ini terdiri dari sebuah Class `NaiveBayesCluster`. Class ini merupakan implementasi Interface `ta.base.NaiveBayes`. Class ini digunakan untuk melakukan klasifikasi dokumen dengan menggunakan algoritma *Naïve Bayes* untuk klaster-klaster yang dihasilkan oleh algoritma *Divisive Information-Theoretic Feature Clustering*.

4.3.2.5. Package Preprocessing

Package preprocessing digunakan untuk melakukan proses *collecting* dan pengindeksan dokumen teks untuk mendapatkan *term-term* dokumen. Dari package ini akan diperoleh *term-term* tiap-tiap dokumen beserta frekuensi kemunculannya tiap-tiap dokumen.



Gambar 4.14. Class diagram package ta.preprocessing

Penjelasan untuk masing-masing Class adalah sebagai berikut:

- Class `StopWords`

Abstract Class yang berfungsi mengecek sebuah *token* termasuk stopword atau bukan. *Method* `isContain()` digunakan untuk mengetahui apakah sebuah *token* merupakan *stopword* atau tidak. *Method* `newInstance()` digunakan untuk mendapatkan instance dari *class* ini.

- Class `StopWordImpl`

Class ini merupakan implementasi dari Class `StopWord`.

- Class `WordList`

Class ini berfungsi untuk menyimpan *term-term* hasil pengindeksan dokumen oleh Class `FileDocument`. *Term-term* tersebut disimpan bersamaan dengan frekuensi kemunculan *term-term* pada dokumen.

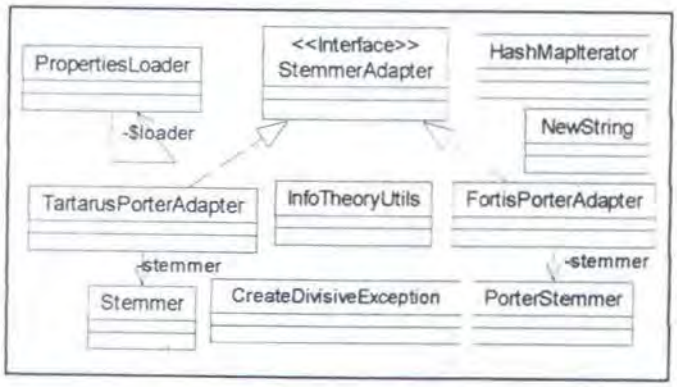
- Class `FileDocument`

Ektraksi (pengindeksan) *token-token* yang terdapat sebuah dokumen adalah fungsi utama dari class ini. *Method* `getTokens()` digunakan untuk mendapatkan *term-term* yang ada pada sebuah dokumen.

Beberapa setting parameter yang digunakan untuk melakukan pengindeksan seperti penggunaan stemmer, pemakaian stopword, konversi *token* akan di ubah ke huruf kecil atau huruf besar, apakah akan menghilangkan *header* dokumen (khusus digunakan pada dataset 20NG) disimpan dalam class ini.

4.3.2.6. Package Utils

Package ini berisi Class-Class yang merupakan *utility Class*. Fungsi-fungsi yang dimiliki oleh Class-Class ini digunakan oleh Class lain.



Gambar 4.15 Class diagram package ta.utils

Class diagram untuk *package* ini ditunjukkan pada gambar 4.13. Penjelasan masing-masing untuk Class yang terdapat pada *package* ini adalah sebagai berikut:

- Class `PropertiesLoader`

Class ini digunakan sebagai *loader Class* untuk file ta.properties. File ini berisi beberapa nilai setting parameter yang digunakan dalam aplikasi ini. File properties diletakan pada *root* dari *classpath* dan diakses pada awal aplikasi dijalankan.

- Class `HashMapIterator`

Iterator Class untuk `HashMap`. *Java API* tidak menyediakan *iterator* untuk Class `java.util.HashMap`. Iterasi komponen-komponen yang terdapat didalam `HashMap` tidak dijamin urut berdasarkan urutan *entry* komponen `Map`. Iterasi dapat dilakukan berdasarkan *key* atau *value*.

- Class `CreateDivisiveException`

Class ini digunakan sebagai `Exception` jika proses pembuatan `CClass` untuk Algoritma Divisive yang dilakukan oleh class `DivisiveFactory` gagal. Class ini merupakan turunan dari Class `java.lang.Exception`.

- Class `InfoTheoryUtils`

Class digunakan melakukan perhitungan yang ada pada algoritma *Divisive Information Theroretic Feature Clustering*. Class ini berisi implementasi program dari rumus-rumus yang digunakan dalam teori informasi.

- Class `PorterStemmer`

Class ini digunakan sebagai implementasi dari algoritma Porter Stemmer. Class ini ditulis oleh Fortis Lazaridis yang dikembangkan dari versi yang ditulis dalam bahasa C.

- Class `Stemmer`

Class ini juga digunakan sebagai implementasi dari algoritma Porter Stemmer. Class ini ditulis oleh para peneliti yang mengembangkan algoritma Porter Stemmer disitus *http://www.tartarus.org/~martin/PorterStemmer*.

- Interface StemmerAdapter

Interface ini digunakan sebagai adapter untuk stemmer yang digunakan. Interface ini diperlukan karena dalam tugas akhir ini ada dua jenis class yang mengimplementasikan algoritma Porter dimana masing-masing class tersebut mempunyai *method* yang tidak sama didalam melakukan steming. Class yang mengimplementasikan interface ini dapat digunakan sebagai stemmer oleh aplikasi. Interface ini hanya memiliki satu *method* `stem()` yang mengembalikan `String` dari token yang sudah di-*stem*. *Method* ini mempunyai parameter `String` *token* yang akan di-*stem*.

- Class TartarusPorterAdapter

Class ini digunakan sebagai *adapter* class yang menjembatani antara class Stemmer agar bisa digunakan sebagai *stemmer* oleh aplikasi. Class ini mengimplementasikan interface StemmerAdapter sehingga *instance* dari class ini yang akan digunakan untuk menjembatani aplikasi untuk menggunakan Porter Stemer dengan class Stemmer.

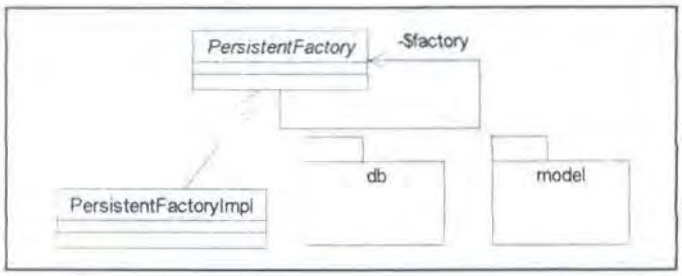
- Class FortisPorterAdapter

Class ini juga digunakan sebagai class *adapter* untuk class PorterStemmer. Sama halnya dengan class TartarusPorterAdapter, *instance* dari class ini bisa digunakan aplikasi untuk melakukan *stemming* sebuah *token*.

4.3.3. Data Layer

Data Layer berfungsi sebagai *layer* yang digunakan untuk melakukan hubungan dengan data-data yang berada diluar sistem aplikasi. Data-data tersebut

disimpan dalam database ataupun dalam file-file eksternal. Secara keseluruhan *data layer* dikelompokan dalam *package* `ta.base.persistent`.



Gambar 4.16 Class diagram untuk data layer.

4.3.3.1. Package Persistent

Persistent package berisi class-class yang berfungsi melakukan persistent data dan *loading* data dari database. *Package* ini terdiri dari dua sub *package* yaitu *packege* `db` dan `model`.:

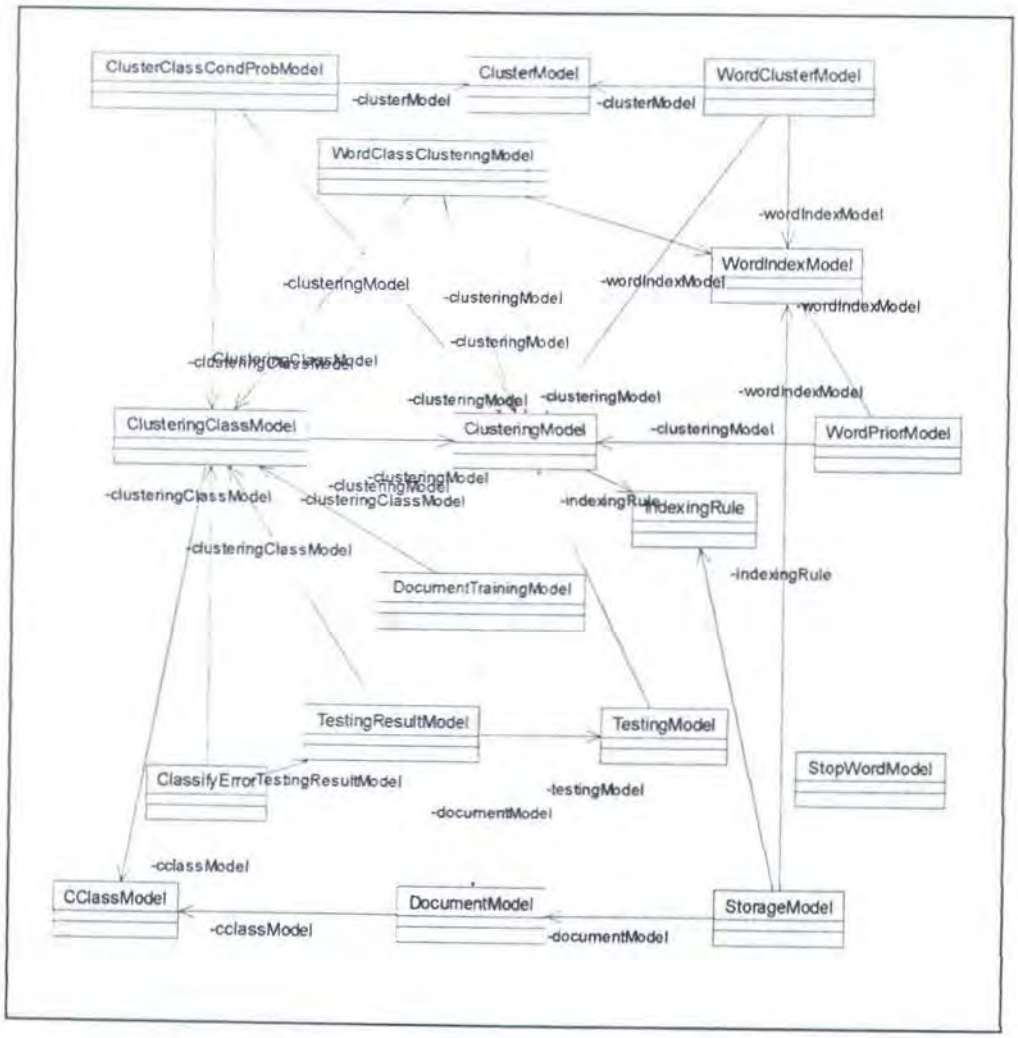
- Class `PersistentFactory`

Class ini adalah *abstract class* yang digunakan sebagai *factory method* untuk setiap class yang berada pada *package* `ta.base.persistent.db`. Berbagai *method* disediakan untuk membuat *instance* dari tiap class yang berada pada *package* `ta.base.persistent.db`. *method* `newInstance()` yang digunakan untuk mendapatkan *instance* dari class ini.

- Class `PersistentFactoryImpl`

Class ini adalah *concrete class* dari *abstract class* `ta.base.persistent.PersistentFactory`.

4.3.3.2. Package Model



Gambar 4.17 Class diagram untuk package ta.base.persistent.model

Class-Class didalam *package* ini adalah representasi *Persistent Class* dari table-table database. Field-field tabel dalam database menjadi properti dari tiap class dengan akses *modifier private*. Untuk mengakses properti digunakan pasangan *method* `setXXX()` dan `getXXX()`, dimana XXX merepresentasikan nama field *properti* pada *class* tersebut. Teknik pemrograman ini juga dikenal dengan nama model pemrograman *Plain Old Java Object* (POJO). Penjelasan masing-masing class yang ada didalam *package* ini adalah sebagai berikut:

- Class `StopWordModel`

Class ini digunakan sebagai representasi entitas dari tabel `stopword`. Objek class ini berisi sebuah field String yang berisi sebuah *stopword*.

- Class `ClusteringModel`

Class ini digunakan sebagai representasi entitas dari table clustering. Class ini berisi setting nilai yang digunakan untuk membentuk klaster. Properti-properti yang digunakan untuk membentuk klaster yaitu jumlah klaster yang akan dibentuk, jumlah kategori yang digunakan dalam klaster, nilai selisih minimum fungsi objectif yang digunakan untuk menghentikan iterasi, list kategori (*document label*) yang digunakan (direpresentasikan dalam class `ClusteringClassModel`) , serta list klaster yang nanti akan dihasilkan (direpresentasikan dalam class `ClusterModel`).

- Class `CClassModel`

Class ini digunakan sebagai reprensentasi entitas dari tabel `cclass`. Class ini berisi nama kategori dari tiap-tiap dokumen. Setiap dokumen hanya mempunyai satu kategori.

- Class `ClusterModel`

Class ini digunakan sebagai representasi entitas dari tabel `cluster`. Class ini berisi nama klaster, nilai *cluster prior* serta list dari *term/word* yang termasuk dalam klaster ini.

- Class `ClusterClassCondProbModel`

class ini digunakan sebagai representasi entitas dari tabel `cluster_class_condprobs`. Class ini berisi *conditional probability* ($P(C|Wj)$

dari klaster untuk masing-masing class `CClassModel`. Class ini mempunyai properti conditionalProb yang digunakan untuk menyimpan nilai *P*(C|*Wj*).

- Class `ClusteringClassModel`

Class ini digunakan sebagai representasi entitas dari tabel `clustering_class`. Objek dari class ini mempunyai properti berupa class yang digunakan dalam klastering.

- Class `DocumentModel`

Class ini digunakan sebagai representasi entitas dari tabel `document`. Objek class ini adalah *classified document* yang merupakan input dari aplikasi ini. Properti dari objek class ini adalah nama dokumen, kategori dokumen, path file dimana dokumen ini berada dan list dari *term* (berupa objek dari class `StorageModel`).

- Class `DocumentTrainingModel`

Class ini digunakan sebagai representasi dari Dokumen yang dipakai sebagai *training dataset* dalam membentuk klaster.

- Class `IndexingRule`

Class ini digunakan sebagai representasi entitas dari tabel `indexing_rule`. Properti yang dimiliki oleh class ini digunakan sebagai parameter untuk melakukan pengindeksan dokumen. Parameter-parameter tersebut adalah *stem* (apakah akan menggunakan stemming), *remove header* (apakah *header* dari dokumen akan dihilangkan), *pruning* (apakah akan menghilangkan kata yang hanya muncul pada beberapa dokumen), *pruning*

occurrence (jumlah maksimal dokumen dimana *term* yang di-*pruning* tersebut muncul).

- Class `StorageModel`

Class ini digunakan sebagai representasi entitas dari tabel `storage`. Objek dari class ini memiliki properti berupa *term* yang terdapat dalam dokumen beserta frekuensi kemunculannya pada dokumen tersebut.

- Class `TestingModel`

Class ini digunakan sebagai representasi entitas pada tabel `testing`. Nilai setting parameter pengujian klaster disimpan dalam class ini. Parameter tersebut adalah jumlah dokumen dari setiap kategori yang digunakan dalam pengujian. Selain itu objek class ini juga menyimpan nilai hasil pengujian untuk masing-masing kategori yang diuji dalam bentuk `List` dari objek class `TestingResultModel`).

- Class `TestingResultModel`

Class ini digunakan sebagai representasi entitas dari tabel `testing_result`. *Instance* class ini berisi hasil pengujian untuk masing-masing kategori. Properti yang dimiliki class ini antara lain adalah jumlah dokumen tiap kategori yang tepat terklasifikasi sesuai dengan kategori dan jumlah dokumen yang salah. Selain itu juga memiliki `List` objek class `ClassifyErrorModel`. Penjelasan mengenai class `ClassifyErrorModel` ada dibagian lain bab ini.

- Classs WordClassClusteringModel

Class ini digunakan sebagai representasi entitas dari tabel `word_class_clustering`. *Instance* class ini merupakan entitas dari probabilitas

kondisional (*conditional probability*) *term* yang dimiliki tiap kategori ($P(C|wt)$) serta frekuensi kemunculan *term* tersebut pada kategori tersebut.

- Class `WordClusterModel`

  Class ini digunakan sebagai representasi entitas dari tabel `word_cluster`.

- Class `WordIndexModel`

  Class ini digunakan sebagai representasi entitas dari table `word_index`.

- Class `WordPriorModel`

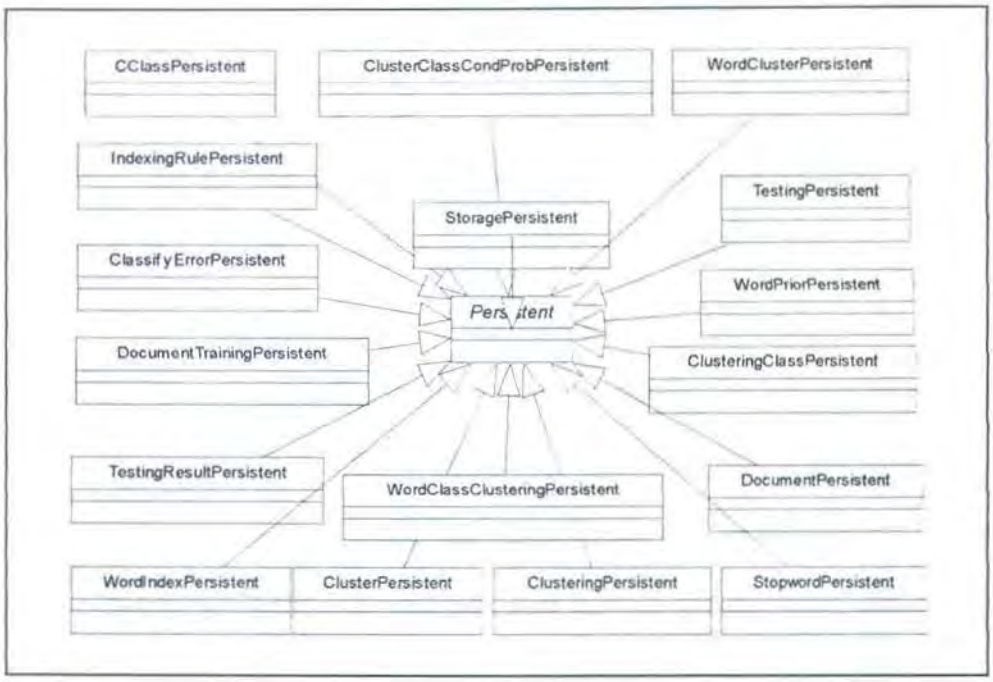
  Class ini digunakan sebagai representasi entitas dari tabel `word_prior`. *Instance-instance* dari class ini adalah sekumpulan *prior probability* ($P(c)$) dari tiap-tiap *term* yang digunakan dalam membentuk klaster.

- Class `ClassifyErrorModel`

  Class ini digunakan sebagai representasi entitas dari tabel `classify_error`. *Property instance* dari *class* ini berisi tentang jumlah dokumen yang salah terklasifikasi pada kategori yang lain

4.3.3.3. Package Db

Package ini berisi class-class yang digunakan untuk melakukan manipulasi data-data tabel yang ada didatabase. Masing-masing *Class-class* digunakan untuk melakukan operasi database *create, insert, update, delete* (CRUD) dan *query* (*loading object*) dari masing-masing table. Setiap *class* yang berada pada *package ta.base.persistent.model* memiliki satu class pada *package* ini yang berfungsi untuk melakukan operasi database.



Gambar 3.18 Class diagram package ta.base.persistent.db

Pembuatan setiap class didasarkan karena ouput yang dihasilkan dari *query* oleh *class hibernate.Session* berupa *instance class Object* sehingga dengan menggunakan satu *class* yang spesifik untuk tiap *data model* output *query* langsung berupa *class* dari *data model* tersebut tanpa harus melalui *type casting*.

Setiap class yang ada pada *package* ini adalah turunan dari *abstract class* `Persistent`. Output yang dihasilkan dari Penjelasan masing-masing class adalah sebagai berikut:

- Class `Persistent`

Abstract class dari masing-masing *class* yang terdapat pada *package* ini. *Method* dasar yang digunakan untuk melakukan operasi CRUD dari *data model* (database) ke aplikasi disediakan oleh class ini. Selain itu *setup* `SessionFactory` dilakukan pada *class* ini.

- Class `StoragePersistent`

Digunakan untuk operasi CRUD dan *query* pada tabel `storage`. Output yang dihasilkan dari *query* adalah objek-objek *instance class* `StorageModel`.

- Class `CClassPersistent`

Digunakan untuk operasi CRUD dan *query* pada tabel `cclass`. Output yang dihasilkan dari *query* adalah objek-objek *instance class* `CClassModel`.

- Class `ClusterClassCondProbPersistent`

Digunakan untuk operasi CRUD dan *query* pada tabel `cluster_class_condprobs`. Output yang dihasilkan dari *query* adalah objek-objek *instance class* `ClusterClassCondProbModel`.

- `ClassWordClusterPersistent`

Digunakan untuk operasi CRUD dan *query* pada tabel `word_cluster`. Output yang dihasilkan dari *query* adalah objek-objek *instance class* `WordClusterModel`.

- Class `IndexingRulePersistent`

Digunakan untuk operasi CRUD dan *query* pada tabel `indexing_rule`. Output yang dihasilkan dari *query* adalah objek-objek *instance class* `IndexingRule`.

- Class `ClassifyErrorPersistent`

Digunakan untuk operasi CRUD dan *query* pada tabel `classify_error`. Output yang dihasilkan dari *query* adalah objek-objek *instance class* `ClassifyErrorModel`.

- Class `DocumentTrainingPersistent`

Digunakan untuk operasi CRUD dan *query* pada tabel `document_training`. Output yang dihasilkan dari *query* adalah objek-objek *instance class* `DocumentTrainingModel`.

- Class `TestingResultPersistent`

Digunakan untuk operasi CRUD dan *query* pada tabel `testing_result`. Output yang dihasilkan dari *query* adalah objek-objek *instance class* `TestingResultModel`.

- Class `WordIndexPersistent`

Digunakan untuk operasi CRUD dan *query* pada tabel `word_index`. Output yang dihasilkan dari *query* adalah objek-objek *instance class* `WordIndexModel`.

- Class `WordClassClusterClusteringPersistent`

Digunakan untuk operasi CRUD dan *query* pada tabel `word_class_clustering`. Output yang dihasilkan dari *query* adalah objek-objek *instance class* `WordClassClusteringModel`.

- Class `ClusterPersistent`

Digunakan untuk operasi CRUD dan *query* pada tabel `cluster`. Output yang dihasilkan dari *query* adalah objek-objek *instance class* `ClusterModel`.

- Class `ClusteringPersistent`

Digunakan untuk operasi CRUD dan *query* pada tabel `clustering`. Output yang dihasilkan dari *query* adalah objek-objek *instance class* `ClusteringModel`.

- Class `StopwordPersistent`

Digunakan untuk operasi CRUD dan *query* pada tabel `stopword`. Output yang dihasilkan dari *query* adalah objek-objek *instance class* `StopWordModel`.

- Class `DocumentPersistent`

Digunakan untuk operasi CRUD dan *query* pada tabel `document`. Output yang dihasilkan dari *query* adalah objek-objek *instance class* `DocumentModel`.

- Class `ClusteringClassPersistent`

Digunakan untuk operasi CRUD dan *query* pada tabel `clustering_class`. Output yang dihasilkan dari *query* adalah objek-objek *instance class* `ClusteringClassModel`.

- Class `WordPriorPersistent`

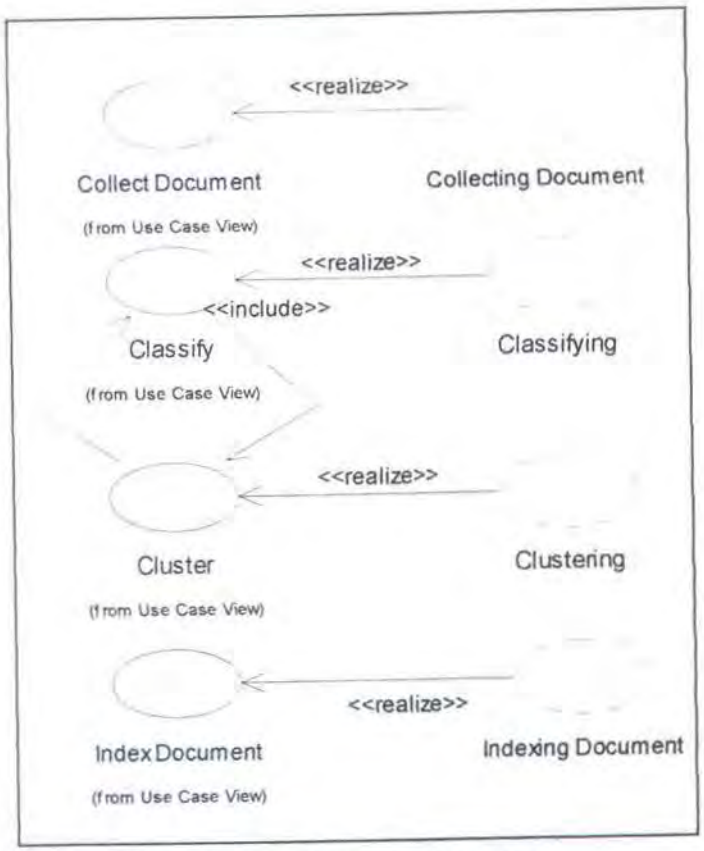
Digunakan untuk operasi CRUD dan *query* pada tabel `word_prior`. Output yang dihasilkan dari *query* adalah objek-objek *instance class* `WordPriorModel`.

- Class `TestingPersistent`

Digunakan untuk operasi CRUD dan *query* pada tabel `testing`. Output yang dihasilkan dari *query* adalah objek-objek *instance class* `TestingModel`.

4.3.4. Use Cases Realization

Use cases realization (UCR) adalah diagram urutan kejadian (*event*). UCR juga sebuah skenario atau realisasi (*instance*) dari *use case* [Ratio-2000]. UCR biasanya berupa *sequence diagram* atau *collaboration diagram*.



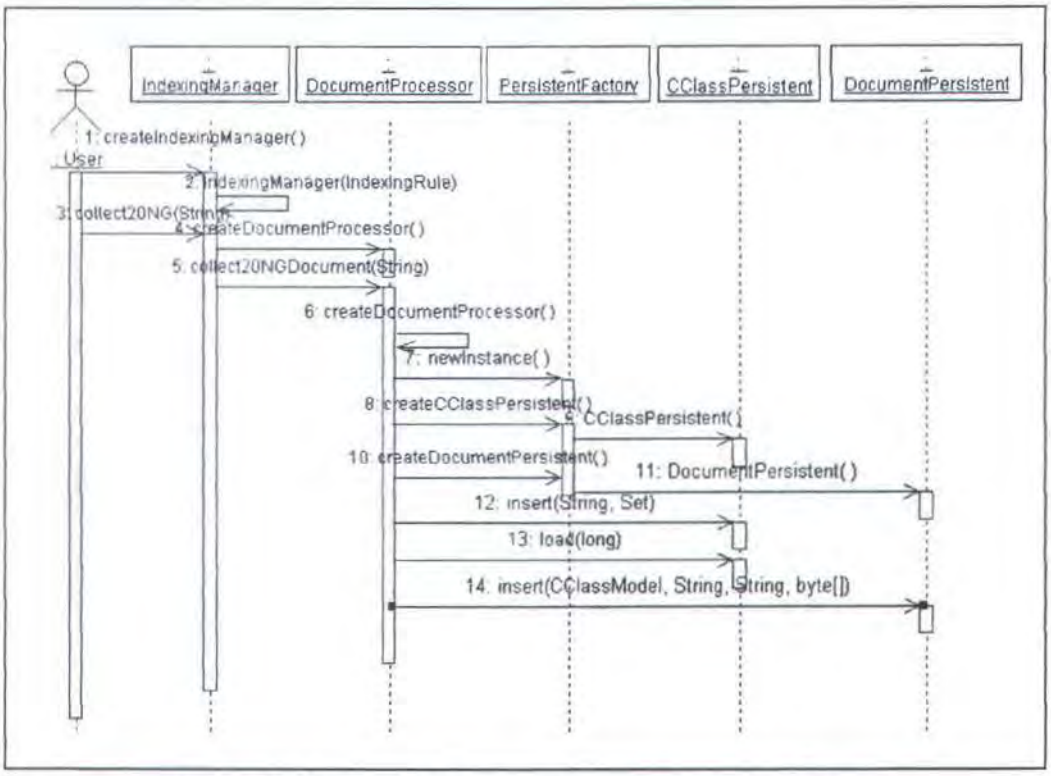
Gambar 4.18 Use cases realization diagram

Sequence diagram adalah gambaran secara grafis dari sebuah skenario yang digunakan untuk menunjukkan interaksi antara obyek atau class dalam sebuah urutan waktu [Ratio-2000]. *Collaboration diagram* digunakan sebagai gambaran grafis dari sebuah diagram yang menunjukkan urutan *message* dari interaksi antara objek atau class didalam mengimplementasikan suatu operasi atau transaksi.

4.3.4.1. Collecting Document

Sequence diagram untuk proses pengumpulan dokumen ditunjukan pada Gambar 4.19. Ada lima class yang berperan dalam proses *collecting document* ini yaitu class `IndexingManager`, class `DocumentPersistent`, class

`PersistentFactory`, class `CClassPersistent` dan class `DocumentPersistent`. Hal pertama kali dilakukan adalah meng-*instantiate* class `IndexingManager` *method* `createIndexingManager()`, dari class ini kemudian dipanggil method `collect20NG()`.



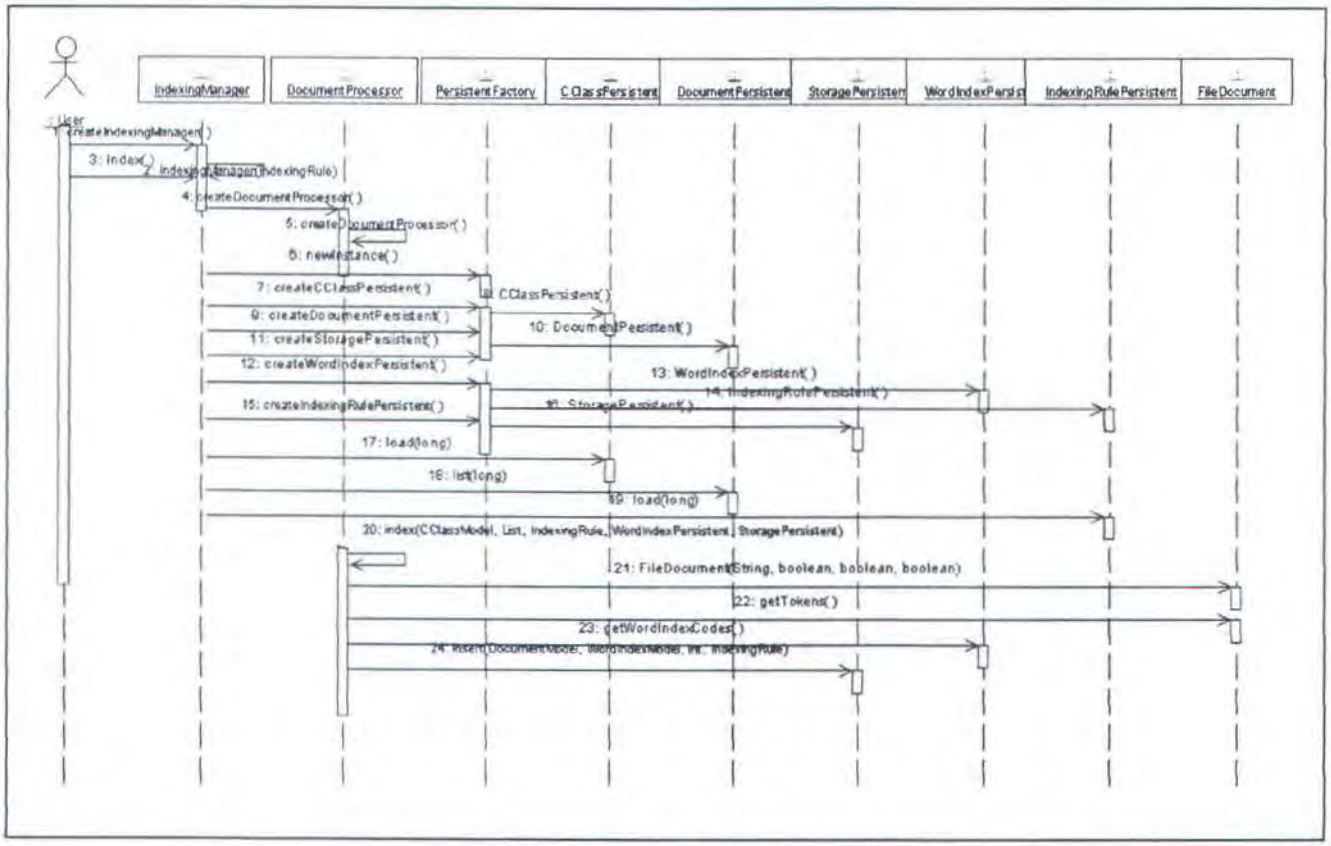
Gambar 4.19 Sequence Diagram untuk proses pengumpulan dokumen

Instance dari class `DocumentProcessor` dibuat dari Class `IndexingManager`. Class `DocumentProcessor` digunakan untuk menggumpulkan file. Informasi dari setiap dokumen disimpan ke datastore oleh class `DocumentPersistent`.

4.3.4.2. Indexing Document

Sequence diagram untuk proses *indexing document* ditunjukan pada gambar 4.20. Class `DocumentPersistent` dipanggil oleh class

`IndexingManager` untuk melakukan proses pengindeksan. Class yang paling utama dalam proses *indexing document* ini adalah class `FileDocument`. Ekstraksi token-token dalam dokumen untuk dipilih menjadi sekumpulan *term-term* dilakukan oleh class ini.

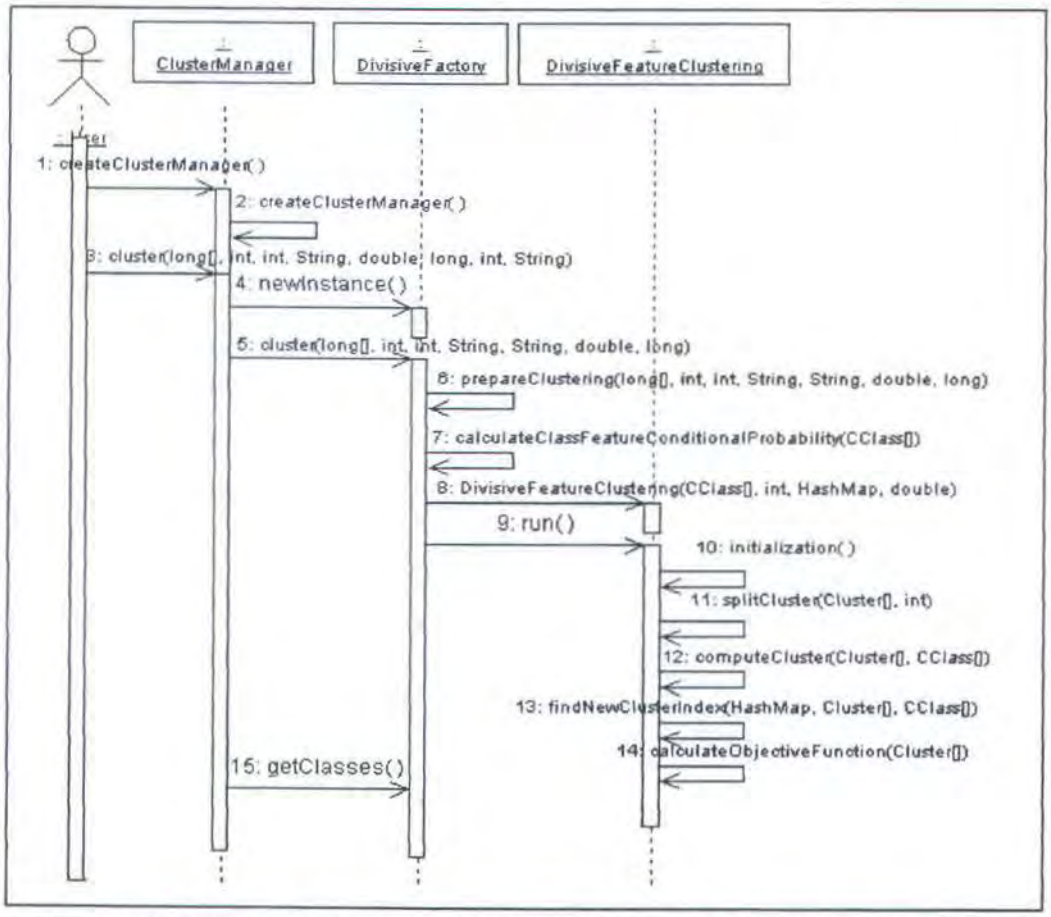


Gambar 4.20 Sequence Diagram proses indexing document.

4.3.4.3. Clustering Document

Sequence diagram yang ditunjukan pada gambar 4.21, digunakan untuk pembentukan *word cluster*. Langkah-langkah yang digunakan untuk membentuk *word cluster* pertama kali adalah class `ClusterManager` di-*instantiate* untuk membentuk *word cluster* dari beberapa class (*document label*) dengan beberapa parameter yaitu jumlah dokumen *training* tiap class, jumlah *word cluster* yang

diinginkan, nilai minumum selisih fungsi objektif untuk menghentikan iterasi, parameter yang digunakan untuk melakukan *indexing* pada dataset seperti penggunaan *stemming* dsb.



Gambar 4.21 Sequence Diagram dari proses pembentukan Word Cluster menggunakan Algoritma Divisive Information Theoretic Feature Clustering

Instance dari class `DivisiveFactory` selanjutnya dibuat oleh class `ClusterManager`. *Method* `prepareClustering()` dijalankan untuk mengumpulkan *term-term* tiap-tiap dokumen sebelum proses klastering dilakukan. *Term-term* tersebut kemudian dihitung probabilitas kondisionalnya dengan tiap-tiap class ($p(C|w_t)$), selain itu *prior probability* dari tiap-tiap *term* juga dihitung.

Dari Class `DivisiveFactory` kemudiaan dibuat *instance* dari class `DivisiveFeatureClustering`.. Dari *instance* class ini kemudian dijalankan *method* `run()` yang berfungsi melakukan proses pembentukan *word cluster*..

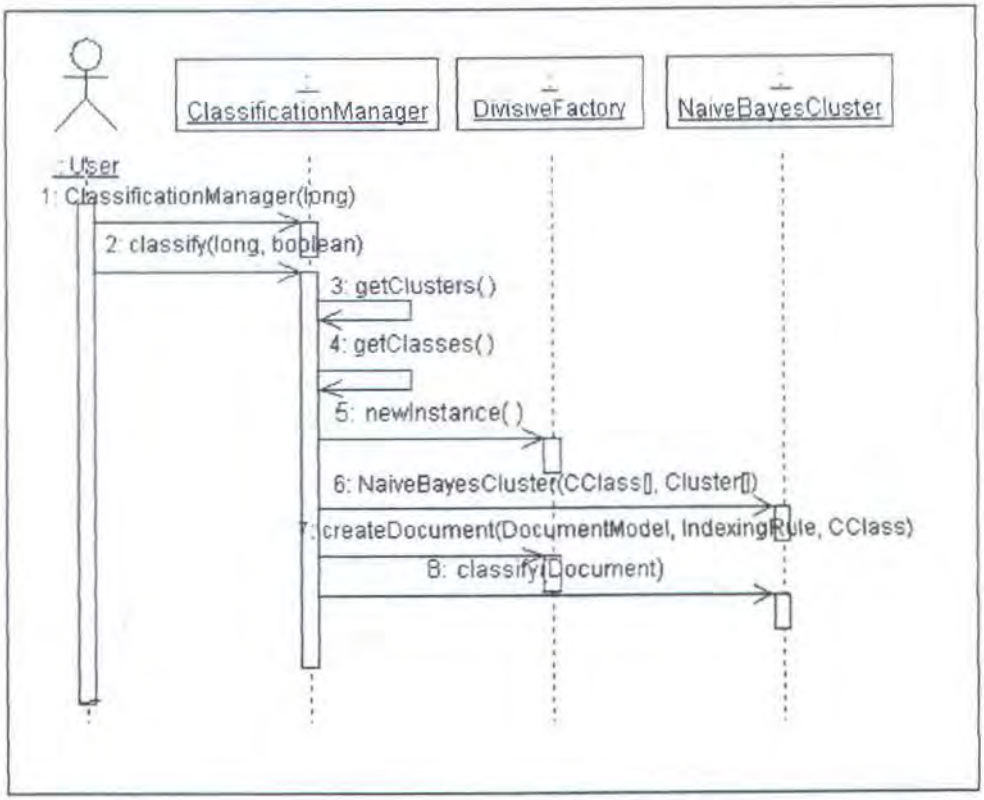
Pada *method* `run()`, terdapat beberapa tahapan yang digunakan untuk membentuk *word cluster*. *method* `initialization()` digunakan untuk membentuk *word cluster* dengan jumlah yang sama dengan jumlah class. *Method* `splitCluster()` digunakan untuk memecah *word cluster* yang dihasilkan dari *method* `initialization()` jika jumlah klaster yang diinginkan tidak sama dengan jumlah class yang digunakan.

Selanjutnya akan dijalankan *method* `computeCluster()`, *method* ini melakukan perhitungan *prior probability* ($p(W)$) dan *word cluster-class conditional probability* ($p(C|W_j)$) untuk tiap-tiap *word cluster*. *Method* `findNewClusterIndex()` digunakan untuk menentukan ulang *word cluster* yang tepatt dari setiap *term*. *Method* `calculateObjectiveFunction()` digunakan untuk menghitung fungsi objektif dari keseluruhan *word cluster*. *Method* `computeCluster()`, *method* `findNewClusterIndex()` dan *method* `calculateObjectiveFunction()` akan dilakukan secara terus-menerus hingga didapatkan nilai selisih fungsi objektif yang sesuai dengan batasan yang diberikan pada parameter.

4.3.4.4. Classify Document

Sequence Diagram pada gambar 4.22 menunjukan alur yang dilakukan dalam melakukan klasifikasi dokumen. Class `NaiveBayesCluster` adalah class yang bertugas untuk melakukan klasifikasi dokumen. Pembuatan *instance* dari

class ini dilakukan oleh class `ClassificationManager`. Konstruktor dari class `NaiveBayesCluster` memerlukan parameter class dari dokumen dan *word cluster* yang berasal dari dokumen *training* masing-masing class tersebut.



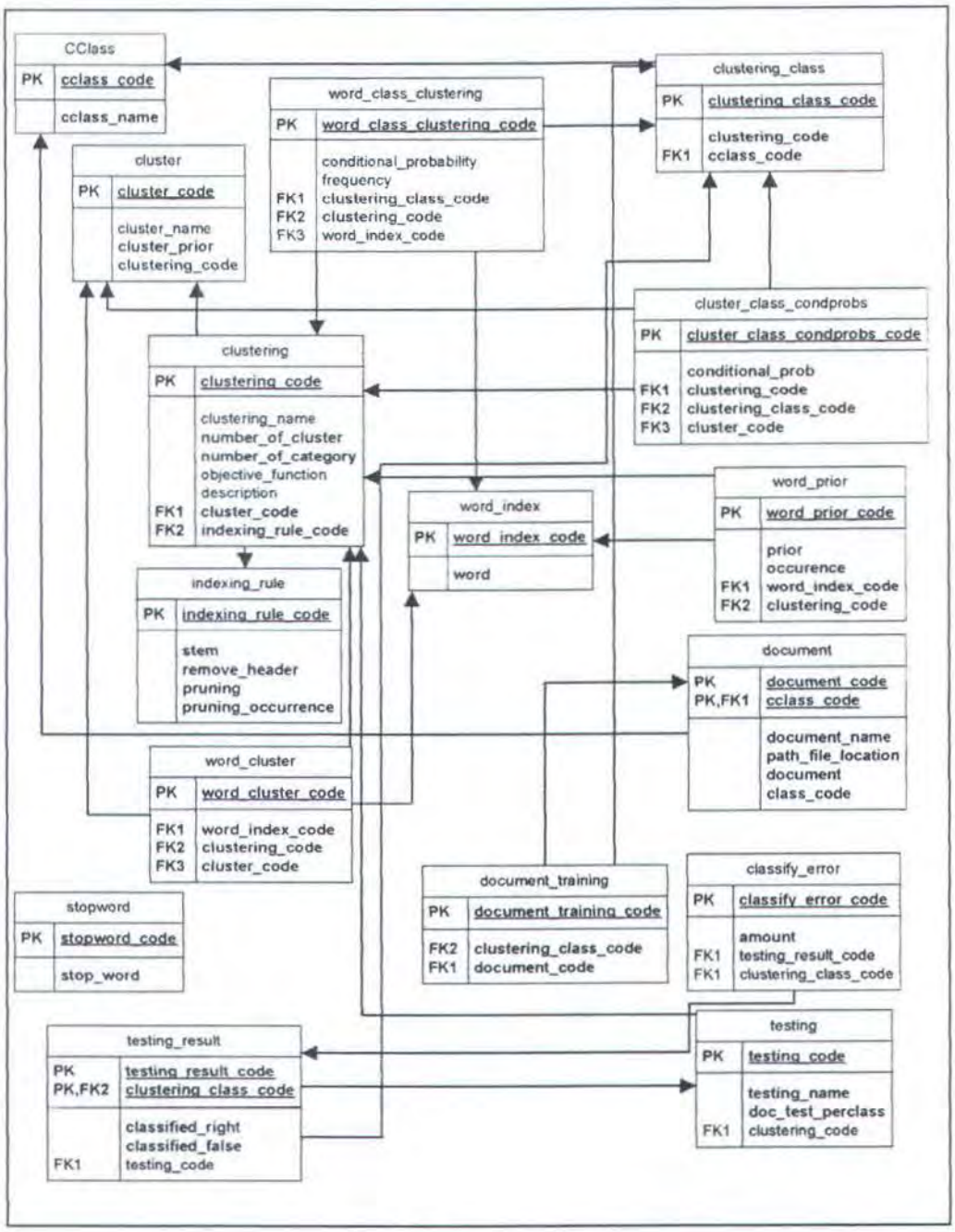
Gambar 4.22 Sequence diagram dari proses klasifikasi dokumen dengan menggunakan *word cluster*.

Klasifikasi dari sebuah dokumen dijalankan dengan memanggil *method* `classify()` pada class `NaivebayesCluster`. Nilai probabilitas dokumen untuk setiap class dikembalikan oleh *method* ini. Nilai probabilitas yang paling besar menunjukan class yang paling mungkin untuk dokumen tersebut.

4.3.5 Perancangan Database

Berdasarkan proses-proses diatas dapat dibuat diagram class entitas yang digunakan sebagai tabel untuk menyimpan data-data yang digunakan dalam

proses-proses diatas. Tabel-tabel yang digunakan dalam tugas akhir ini ditunjukkan pada gambar 4.23.



Gambar 4.23. Diagram ERD aplikasi

BAB V
IMPLEMENTASI PERANGKAT LUNAK

Pada bab ini akan dijelaskan fungsi dan kegunaan *method-method* yang dimiliki beberapa class penting, yang diperlukan dalam proses inti aplikasi.

5.1. APPLICATION LAYER

Tahap implementasi perangkat lunak dilakukan setelah tahap perancangan selesai. Pada pelaksanaanya terdapat proses dimana desain yang telah dibuat diubah kembali agar sesuai dengan program. Class-class yang telah didefinisikan pada tahap desain akan diterapkan pada bahasa pemrograman Java.

Class-class yang termasuk dalam *application layer* adalah class-class yang berinteraksi langsung dengan pemakai aplikasi. Class-class ini mempunyai antarmuka dan menerima action-action yang dilakukan *user*, untuk kemudian memanggil *method* yang sesuai.

5.1.1. Ta

```
public class Ta {
  boolean packFrame = false;

  public Ta() {}

  public static void main(String[] args) {}
}
```

Gambar 5.1 Fungsi-fungsi class Ta

Class ini digunakan untuk menjalankan aplikasi ini. Class ini memiliki *method* `main()` yang digunakan sebagai main class untuk aplikasi.

5.1.2. TAFrame

```
public class TAFrame
    extends JFrame {

  public TAFrame() {}

  private void jbInit() throws Exception {}

  void jMenuFileExit_actionPerformed(ActionEvent actionEvent) {}

  void jMenuHelpAbout_actionPerformed(ActionEvent actionEvent) {}
  ...
}
```

Gambar 5.2 Fungsi-fungsi class TAFrame

Class ini digunakan sebagai class utama dari *application layer*. class ini menyediakan interface yang digunakan oleh *user* berinteraksi dengan aplikasi. *Method* `jbInit()` digunakan untuk melakukan inisialisasi dari komponen-komponen yang membentuk user interface ini.

5.1.3. IndexPanel

```
public class IndexPanel extends JPanel {
  public IndexPanel() {}

  private void jbInit() throws Exception {}

  public void showDocument(long cclassCode){}

  public void showDocumentFeature(long documentCode){ }

  public void indexDocument(boolean stem, boolean
      removeHeader, boolean userStemmer, StemmerAdapter
      adapter){}

  public void refreshGUI(){}
  ...
}
```

Gambar 5.3 Fungsi-fungsi class IndexPanel

Class ini digunakan oleh aplikasi untuk melakukan pengindeksan dan menampilkan *term-term* dari dokumen. *Method* `showDocument()` digunakan untuk menampilkan daftar dokumen untuk kategori/class tertentu. *Method* `showDocumentFeature()` digunakan untuk menampilkan daftar *feature* dari

sebuah dokumen. Sedangkan *method* `indexDocument()` digunakan untuk melakukan pengindeksan dokumen. *Method* `refreshGUI()` digunakan untuk melakukan refreshing panel jika ada perubahan data.

5.1.4. ClusterPanel

```
public class ClusterPanel extends JPanel {
  public ClusterPanel() {}

  private void jbInit() throws Exception {}

  public void showCluster(long clusteringCode){}

  public void createCluster(){}

  public void deleteCluster(long clusteringCode){}

  protected void refreshGUI(){}
  ...
}
```

Gambar 5.4 Fungsi-fungsi class ClusterPanel

Class ini digunakan untuk menampilkan hasil klastering serta sebagai *form* untuk melakukan klastering. *Method* `showCluster()` digunakan untuk menampilkan data-data hasil klastering, sedangkan *method* `createCluster()` digunakan untuk membuat *word cluster*. *Method* `deleteCluster()` digunakan untuk menghapus *word cluster* yang sudah dihasilkan apabila dibutuhkan. Sedangkan *method* `refreshGUI()` digunakan untuk melakukan *refershing* `ClusterPanel`.

5.1.5. ClassifyPanel

Class ini digunakan untuk menampilkan hasil klasifikasi dokumen. *Method* `classify()` digunakan untuk melakukan klasifikasi dari sejumlah dokumen. Hasil klasifkasi ini bisa di lihat dengan memanggil *method*

`showClassificationResult()`. *Method* `refreshGUI()` digunakan untuk merefresh tampilan dari panel ini.

```
public class ClassifyPanel extends JPanel {
  public ClassifyPanel() {}

  private void jbInit() throws Exception {}

  public void classify(long clusteringCode, int
numberOfDocumentTest){}

  public void refreshGUI(){}

  public void showClassificationResult(long testCode){}
  . . .
}
```

Gambar 5.5 Fungsi-fungsi class ClassifyPanel

5.2. BUSSINESS LAYER

Class-class yang termasuk dalam *bussiness layer* digunakan untuk melakukan pengolahan atau pemrosesan pada data-data yang didapat dari *application layer* atau *data layer*. Class-class dan Interface ini juga berfungsi sebagai penghubung antara *application layer* dan *data layer*.

5.2.1. Feature

```
package ta.base;

public interface Feature
{

   public void setFeatureCode(long featureCode);

   public void setFrequency(int frequency);

   public void setFeature(String feature);

   public Long getFeatureCode();

   public int getFrequency();

   public String getFeature();
}
```

Gambar 5.6 Fungsi-fungsi dalam Interface Feature

`Interface Feature` adalah `Interface` yang digunakan sebagai *term* dalam algoritma *Divisive Information Theoretic Feature Clustering*. frekuensi dari kemunculan *term* dapat diperoleh melalui *method* `getFrequency()`. Dari method `getFeature()` juga didapatkan *term*. *Method* `getFeatureCode()` digunakan untuk mendapatkan kode dari term pada table.

5.2.2. Document

```
package ta.base;

public interface Document {
   public long getDocumentCode();

   public Feature[] getFeatures();

   public int getFeatureFrequency(long featureCode);

   public int getDocumentLength();

   public String getDocumentTitle();

   public long getDocumentClassCode();
}
```

Gambar 5.7. Fungsi-fungsi dalam interface Document

`Interface Document` adalah representasi dari dokumen yang digunakan untuk klasifikasi. `Interface` ini digunakan sebagai representasi dari dataset yang digunakan untuk *training* dan *testing*. Dari interface ini bisa diperoleh *term* yang ada dengan *method* `getFeatures()`, *method* `getDocumentTitle()` digunakan untuk mendapatkan nama dokumen. Sedangkan *method* `getDocumentClassCode()` digunakan untuk mendapatkan kode kategori dokumen.

5.2.3. CClass

Interface `CClass` digunakan untuk merepresentasikan kategori dokumen dalam algoritma *divisive information theoretic feature clustering*. Interface

`CClass` memiliki *method* `getClassFeatureConditionalProbability()` yang digunakan untuk mendapatkan probabilitas kondisional dari sebuat *term* ($p(C|w_t)$).

```
public interface CClass {
    public double getClassFeatureConditionalProbability(Feature feature);
    public int getFeatureFrequency(Feature feature);
    public void setClassPrior(double classPrior);
    public double getClassPrior();
    public int getClusterFeaturesFrequency(Cluster cluster);
    public void setClassFeatureConditionalProbability(long featureCode,
           int totalFrequency);
    public Feature[] getClassFeatures();
    public void setClassFeatures(Feature[] features);
    public void addFeature(long featureCode, String feature, int frequency);
    public void removeFeature(long featureCode);
    public void setClusterConditionalProbability(Cluster cluster, double value);
    public double getClusterConditionalProbability(Cluster cluster);
    public void addTrainingDocument(Document documents);
    public void setFeatures(HashMap features);
    public HashMap getFeatures();
    public ClusteringClassModel getClusteringClass();
    ...
}
```

Gambar 5.8. Fungsi-fungsi dalam interface CClass

Method `getFeatureFrequency()` digunakan untuk mendapatkan frekuensi kemunculan *feature* dalam dataset dokumen yang digunakan untuk training, sedangkan `method getClusterConditionalProbability()` digunakan untuk mendapatkan probabilitas dari $p(C|W_j)$.

5.2.4. Cluster

Interface Cluster digunakan sebagai klaster dalam algoritma *divisive information theoretic feature clustering*. Fungsi `getClusterPrior()` digunakan untuk mendapatkan nilai probabilitas prior dari klaster. *Method* `calculateClusterPrior()` digunakan untuk menghitung nilai probabilitas prior dari klaster.

```
public interface Cluster {
    public double getClusterPrior();

    public void calculateClusterPrior(HashMap featurePrior);

    public void calculateClassConditionalProbability(CClass class1,
            HashMap featurePrior);

    public void addFeatures(Feature[] features);

    public void removeFeature(Feature feature);

    public List splitFeatures(int spliter);

    public double[] getClassConditionalProbability(CClass[] classes);

    public long getClusterCode();

    public Feature[] getFeatures();

    public double getDocumentConditionalProbability(Document document);

    public double getWordClusterClassCondProb(CClass class1);

    public boolean isContainFeature(Feature feature);
    ....
}
```

Gambar 5.9. Fungsi-fungsi dalam interface Cluster

Sedangkan *method* `calculateClassConditionalProbability()` digunakan untuk menghitung $p(C|W_f)$. Setiap iterasi yang dilakukan oleh algoritma *divisive information theoretic feature clustering* selalu ada *term* baru yang masuk atau keluar dari klaster, untuk itu *method* `addFeatures()` digunakan untuk menambah *term ke* klaster, sedangkan *method* `removeFeature()`

digunakan untuk menghilangkan sebuah *term* yang tidak termasuk dalam klaster. *Method* `getDocumentConditionalProbability()` digunakan untuk mendapatkan probabilitas kondisional dari sebuah dokumen (direpresentasikan dengan interface Document) terhadap klaster ($p(W_s|d)$).

5.2.5. DivisiveFactory

```
public abstract class DivisiveFactory {
    private DivisiveFactory() {}

    public static DivisiveFactory newInstance() { }

    public abstract CClass createCClass(long classCode, long clusteringCode,
            List documentTraining);

    public abstract Cluster createCluster(long clusterCode, long clusteringCode);

    public abstract Feature createFeature(WordIndexModel     wordIndexModel,
            StorageModel storageModel);

    public abstract Document createDocument(DocumentModel model,
            IndexingRule rule, CClass cclass);

    public abstract HashMap getFeaturePrior(long clusteringCode);

    protected abstract CClass[] prepareClustering(long[] cclassCodes,
            int desiredCluster, int docTrainingPerClass, String clusterName,
            String description, double objectiveFunction, long indexingRuleCode);

    protected abstract HashMap calculateClassFeatureConditionalProbability(
            CClass[] classes);

    public synchronized Cluster[] cluster(long[] classCodes,
            int desiredCluster, int docTrainingSizePerClass,
            String clusteringName, String description,
            double objectiveFunction, long indexingRuleCode) { }

    protected abstract void persistClass(CClass cclass,
            PersistentFactory factory);
    public abstract StemmerAdapter getStemmer();
    ....
}
```

Gambar 5.10. Fungsi-fungsi dalam class DivisiveFactory

Class ini merupakan *factory method* yang digunakan untuk membuat *instance* dari beberapa *class*. Class ini merupakan *abstract class*. *Method* `newInstance()` digunakan untuk mendapatkan *concrete* objek dari class ini.

Method ini buat static *instance* sehingga merupakan *class method*. Konstruktor dari *class* dibuat akses *private* sehingga hanya bisa diakses oleh *class* ini sendiri. Dari *method* `newInstance()` dapat ditentukan *instance concrete class* yang merupakan turunan dari *class* DivisiveFactory yang akan dikembalikan oleh *class method* ini.

Method `createCClass()` digunakan untuk membuat *instance* dari class yang menngimplementasikan interface `CClass`. *Method* `createFeature()` digunakan untuk membuat *instance* dari class yang mengimplementasikan interface `Feature`. Sedangkan *method* `createCluster()` digunakan untuk membuat *instance* dari class yang mengimplementasikan interface `Cluster`, method `createDocument()` digunakan untuk membuat *instance* dari class yang mengimplementasikan interface `Document`.

Method `cluster()` digunakan untuk membentuk klaster dari *term* dengan menggunakan algoritma *divisive information theoretic feature clustering*. *Method ini* hanya mem-passing parameter ke class `DivisiveFeatureClustering` yang berfungsi untuk membentuk klaster dari *term*. Sedangkan *method* `getStemmer()` digunakan untuk mendapatkan class yang mengimplementasikan interface `StemmerAdapter`.

5.2.6. NaiveBayes

Interface NaiveBayes adalah interface digunakan untuk yang merepresentasikan algoritma *naïve bayes* untuk pengklasifikasian dokumen. Detail implementasi dari masing-masing *method* yang terdapat dalam interface ini diserahkan ke masing-masing class yang mengimplementasikannya. Pembuatan

interface NaiveBayes disebabkan oleh adanya perbedaan perhitungan yang digunakan untuk melakukan klasifikasi dengan menggunakan *word cluster* dan yang tidak menggunakan *word cluster*.

```
public interface NaiveBayes {
    public void classify(Document document);

    public void setClasses CClass[] classes);

    public CClass[] getClasses();

    public double[] getClassProbs();
}
```

Gambar 5.11. Fungsi-fungsi yang ada dalam interface NaiveBayes

Method `classify()` digunakan untuk menentukan kategori yang paling mungkin untuk sebuah dokumen. Sedangkan *method* `setClasses()` digunakan untuk melakukan pengesetan kategori-kategori dokumen yang digunakan dalam klasifikasi. Parameter berupa array dari class yang mengimplementasikan interface `CClass` diperlukan oleh method ini

5.2.7. NaiveBayesCluster

```
public class NaiveBayesCluster implements NaiveBayes {

    public NaiveBayesCluster(CClass[] classes, Cluster[] clusters) { }

    private void setClassWordClusterCondProbs() {}

    public void classify(Document document) {}

    private void classify(Document document, CClass[] classes,
            Cluster[] clusters) {}

    public double[] getClassProbs() {
    }

    public long getCategory(Document document) {
    }
    ....
}
```

Gambar 5.12 Fungsi-fungsi dalam class NaiveBayesCluster

Class ini digunakan sebagai implementasi interface `NaiveBayes` untuk *word cluster*. *Method-method* yang ada pada interface `NaiveBayes` diimplementasikan dalam class ini dengan menyesuaikan perhitungkan algoritma *naïve bayes* dengan menggunakan *word cluster*.

5.2.8. DivisiveFeatureClustering

```
public class DivisiveFeatureClustering {
    public DivisiveFeatureClustering(CClass[] classes, int numberOfClusters,
            HashMap prior, double changeObjectiveFunction) {}

    public Cluster[] run() {}

    public double getObjectiveFunction() {}

    private double calculateObjectiveFunction(Cluster[] clusters) {}

    private void findNewClusterIndex(HashMap features, Cluster[] clusters,
            CClass[] classes) {}

    private HashMap argmin(double[] featureClassCondProbs, Cluster[]
clusters,CClass[] classes) {}

    private void computeCluster(Cluster[] clusters, CClass[] classes) {}

    private Cluster[] splitCluster(Cluster[] clusters, int desiredCluster){}

    private Cluster[] initialization() {}

    private void assignFeatureMaxClass(Cluster[] clusters, CClass[] classes,
            long featureCode) {}
    ...
}
```

Gambar 5.13 Fungsi-fungsi dari class DivisiveFeatureClustering

Class ini digunakan sebagai implementasi algoritma *divisive information theoretic feature clusetering*. Beberapa parameter yang diperlukan oleh algoritma ini yaitu *feature prior* yang diimplementasikan dengan `HashMap` prior, array dari `CClass` yang digunakan dalam membentuk klaster, jumlah klaster yang akan dibentuk serta nilai selisih fungsi objektif yang digunakan untuk menghentikan iterasi.

Method `run()` digunakan untuk membentuk *word cluster*. *Word cluster* yang dihasilkan olah class ini dikembalikan oleh *method* ini. *Method* `initialization()` digunakan untuk membentuk *initial word cluster*, jumlah *initial word cluster* ini sama dengan jumlah `CClass` yang ikut dalam klaster ini.

Setelah tahapan inisialisasi selesai dilakukan, *method* `splitCluster()` dilakukan untuk memecah *initial word cluster* menjadi sejumlah klaster yang sesuai dengan jumlah *word cluster* yang akan dibentuk. *Method* `computeCluster()` digunakan untuk menghitung *cluster prior* dan probabilitas kondisional *Cluster* terhadap kategori ($p(C|W_j)$). Setelah itu *method* `findNewClusterIndex()` dijalankan untuk menempatkan setiap *term* kedalam klaster yang tepat.

Sebuah *term* termasuk dalam sebuah klaster jika jarak *term* dengan klaster tersebut adalah yang paling minimal diantara *term* dengan klaster-klaster yang lainnya. *Method* `calculateObjectiveFunction()` digunakan untuk mengukur kualitas dari klaster-klaster yang dihasilkan. Jika nilai selisih fungsi objektif yang dihasilkan dengan nilai fungsi objektif iterasi sebelumnya sama dengan atau lebih kecil dari nilai `changeObjectiveFunction`, maka proses pembentukan klaster dihentikan. Jika sebaliknya maka *method* berikut ini akan dijalankan ulang secara berurutan `computeCluster()`, `findNewClusterIndex()` dan `calculateObjectiveFunction()`.

5.2.9. FileDocument

Class ini digunakan untuk melakukan *indexing* dokumen teks. Konstruktor dari class ini menggunakan beberapa parameter yang digunakan untuk melakukan *indexing*. `fullDocumentName` adalah nama file yang akan di indeks.

```
public class FileDocument {
    public static final String TOKENIZERDELIMITER = "
\t\n\r\f\'\"\\1234567890!@#$%^&*()_+-={}|[]:;<,>.?/`~";

    public FileDocument(String fullDocumentName, boolean stem,
            boolean removeHeader, boolean convertToLowerCase,
            StemmerAdapter stemmerAdapter) throws
FileNotFoundException {}

    public WordList getTokens() throws IOException {}

    ...
}
```

Gambar 5.14. Fungsi-fungsi dari class FileDocument

`stem` digunakan untuk mengecek apakah akan dilakukan stemming untuk setiap token atau tidak, `removeHeader` digunakan untuk tidak menyertakan header dokumen yang berasal dataset 20NG, `converToLowerCase` digunakan untuk mengkonversi semua token kedalam huruf kecil semua, dan `stemmerAdapter` adalah *instance* dari class yang mengimplementasikan interface `StemmerAdapter`.

Term-term yang dihasilkan dari file ini bisa diperoleh dengan menggunakan *method* `getToken()`. Class `WordList` berisi daftar setiap *term* beserta frekuensi kemunculannya. Gambar 5.14 adalah contoh artikel yang diambil dari dataset 20 Newsgroups. Dokumen ini berasal dari topik `sci.med`.

```
Newsgroups: sci.med
Path:
cantaloupe.srv.cs.cmu.edu!magnesium.club.cc.cmu.edu!news.sei.
cmu.edu!cis.ohio-state.edu!zaphod.mps.ohio-
state.edu!howland.reston.ans.net!uxl.cso.uiuc.edu!uxl.cts.eiu
.edu!cfaks
From: cfaks@uxl.cts.eiu.edu (Alice Sanders)
Subject: Re: Antihistamine for sleep aid
Message-ID: <1993Apr30.202808.19204@uxl.cts.eiu.edu>
Date: Fri, 30 Apr 1993 20:28:08 GMT
References: <1993Apr29.052044.23918@nmt.edu>
<SDL.93Apr30120345@rigel.linus.mitre.org>
Organization: Eastern Illinois University
Lines: 10

But after you have taken antihistamines for a few nights,
doesn't it start
to have a paradoxical effect?  I used to take one every night
for
allergies and couldn't figure out why I developed bad
insomnia.  Finally
figured out it was the antihistamines.  I would fall asleep
for a few
minutes but would awaken at the drop of a pin a little later
and could not
get back to sleep.  I don't have that problem since I stopped
the
antihistamines at bedtime.  ?

Alice
```

Gambar 5.15. Dokumen yang diambil dari dataset 20 Newsgroups topik sci.med

Hasil pengindeksan yang tanpa menggunakan *stemmer* ditunjukan pada tabel 5.1, sedangkan hasil pengindeksan dengan *stemmer* ditunjukan pada tabel 5.2. Stemmer yang digunakan adalah PorterStemmer. Dari kedua tabel tersebut bisa diketahui bahwa tanpa menggunakan *stemmer token* antihistamine dan antihistamines adalah token yang berbeda, sedangkan jika *stemmer* digunakan maka keduanya digabungkan menjadi satu *token* yaitu antihistamin.

Tabel 5.1 Hasil pengindeksan tanpa menggunakan stemming

Term	Frekuensi	Term	Frekuensi
antihistamine	1	insomnia	1
sleep	2	finally	1
aid	1	figured	1
antihistamines	3	fall	1
nights	1	asleep	1

doesn	1	minutes	1
start	1	awaken	1
paradoxical	1	drop	1
effect	1	pin	1
night	1	back	1
ellergies	1	don	1
couldn	1	problem	1
figure	1	stopped	1
developed	1	bedtime	1
bad	1	alice	1

Table 5.2. Hasil pengindeksan dengan menggunakan Porter Stemmer

Term	Frekuensi	Term	Frekuensi
antihistamin	4	fall	1
sleep	2	asleep	1
aid	1	minut	1
night	2	awaken	1
doesn	1	drop	1
start	1	pin	1
paradox	1	back	1
effect	1	don	1
allergi	1	problem	1
couldn	1	stop	1
figur	2	bedtim	1
develop	1	alic	1
bad	1		
insomnia	1		
final	1		

5.2.10. WordList

```
public class WordList
{
        public void add(String word, int frequency) {}

        public void add(WordList wordList) {}

        public void remove(String word) {}

        public Iterator getKeysIterator(){}

        public int getFrequency(String word) {}

        ...
}
```

Gambar 5.16. Fungsi-fungsi dari Class WordList.

Class ini digunakan untuk menyimpan *term* berserta frekuensi kemunculannya. *Instance* dari class ini adalah nilai yang dikembalikan oleh *method* `getToken()` pada class `FileDocument`. *Method* `add()` digunakan untuk menambahkan sebuah *term*, *method* `remove()` digunakan untuk membuang *term*. *Method* `getKeysIterator()` digunakan untuk mendapatkan Iterator yang digunakan untuk melakukan *iterasi* pada list *term* tersebut, sedangkan *method* `getFrequency()` digunakan untuk mendapatkan frekuensi kemunculan sebuah *term*.

5.2.11. HashMapIterator

Class ini merupakan class Iterator untuk `HashMap`. Class ini dibuat karena Java API tidak menyediakan `Iterator` untuk class `HashMap`. Class ini mengimplementasikan interface `java.util.Iterator`. Class ini sebenarnya merupakan *wrapper* dari iterator property `EntrySet` yang terdapat dalam `HashMap`.

```
public class HashMapIterator implements Iterator {
    public HashMapIterator(HashMap hashMap) {}

    public boolean hasNext() {}

    public Object next() {}

    public Object nextKey() {}

    public boolean hasNextKey() {}

    public void remove() {}
    ...
}
```

Gambar 5.17. Fungsi-fungsi Class HashMapIterator

Selain *method-method* yang terdapat pada interface `Iterator`, class ini juga mempunyai *method* `nextKey()` dan `hasNextKey()` yang digunakan untuk

melakukan iterasi pada *key* property `HashMap`. *Method-method* bawaan dari interface `Iterator` digunakan untuk melakukan iterasi pada *value property* `HashMap`. Iterasi yang dilakukan `HashMapIterator` tidak dilakukan secara urut sesuai dengan urutan masuknya *item* (*key-value*) pada `HashMap`.

5.2.15. InfoTheoryUtils

```
public class InfoTheoryUtils{
    public static final double LOG_BASE_2 = (double) 2.0;

    public static final double K = (double) -1.0;

    public static double log2(double variable) {}

    public static double entropy(double[] distributions) {}

    public static double kullbackLeiblerDivergence(double[]
distributions1,
            double[] distributions2) {}

    public static double jensenShannonDivergence(double[]
distributionWeights,
            double[][] distributions) {}
    ...
}
```

Gambar 5.18. Fungsi-fungsi class InfoTheoryUtils

Class ini digunakan untuk melakukan perhitungan rumus-rumus yang banyak digunakan dalam algoritma *divisive information theoretic feature clustering*. *Method* `entropy()` digunakan untuk menghitung nilai entropy dari sebuah distribusi probabilitas. *Method* `kullbackLeiblerDivergence()` digunakan untuk menghitung jarak antara dua distribusi probabilitas dengan menggunakan rumus Kullback-Leibler divergence. *Method* `jensenShannonDivergence()` masing-masing digunakan untuk menghitung kemiripan dari beberapa distribusi probabilitas. *Method* `log2()` digunakan untuk menghitung logaritmik dengan base 2.

5.3. DATA LAYER

Class-class yang termasuk dalam *data layer* digunakan untuk melakukan penulisan data dari *bussiness layer* dan pembacaan data untuk *bussiness layer*. Aplikasi dengan Hibernate dihubungkan dengan class ini.

5.3.1. PersistentFactory

```
public abstract class PersistentFactory {

    public static PersistentFactory newInstance() {}

    public abstract CClassPersistent createCClassPersistent() throws Exception;

    public abstract ClusteringClassPersistent createClusteringClassPersistent()
            throws Exception;

    public abstract ClusteringPersistent createClusteringPersistent()
            throws Exception;

    public abstract ClusterPersistent createClusterPersistent()
            throws Exception;

    public abstract DocumentPersistent createDocumentPersistent()
            throws Exception;

    public abstract DocumentTrainingPersistent createDocumentTrainingPersistent()
            throws Exception;

    public abstract StoragePersistent createStoragePersistent()
            throws Exception;

    public abstract TestingPersistent createTestingPersistent()
            throws Exception;

    public abstract WordClassClusteringPersistent
createWordClassClusteringPersistent()
            throws Exception;

    public abstract WordClusterPersistent createWordClusterPersistent()
            throws Exception;

    public abstract WordIndexPersistent createWordIndexPersistent()
            throws Exception;

    public abstract WordPriorPersistent createWordPriorPersistent()
            throws Exception;

    public abstract ClusterClassCondProbPersistent
createClusterClassCondProbPersistent()
            throws Exception;
    . . .
}
```

Gambar 5.19 Fungsi-fungsi dalam class PersistentFacory.

Class ini berfungsi sebagai *factory method* yang digunakan untuk membuat *instance* dari beberapa class. Class ini adalah *abstract* class, sedangkan class yang merupakan turunan langsung dari class ini adalah class `PeristentFactoryImpl`. *Method-method* yang ada didalam class digunakan membuat *instance* dari class yang menghubungkan setiap *domain model* dengan Hibernate.

Method `newInstance()` digunakan untuk membuat *singleton instance* dari class ini sehingga hanya ada satu *instance* dalam aplikasi . *Method* ini merupakan *static method* yang mengembalikan *instance* dari class `PersistentFactoryImpl` yang merupakan class turunan dari class `PersistentFactory`. Gambar 5.19 merupakan detail dari *method* `newInstance()`.

```
private static PersistentFactory factory = null;

public static PersistentFactory newInstance() {
        if (null == factory)
            factory = new PersistentFactoryImpl();
        return factory;
    }
```

Gambar 5.20 Method newInstance() untuk membuat Singleton Object class PersistentFactory

Method `createCClassPersistent()` digunakan untuk membuat *instance* dari class `CClassPersistent`. Sedangkan *Method* `createClusteringClassPersistent()` digunakan untuk membuat *instance* dari class `ClusteringClassPersistent` serta *method-method* yang lainnya.

5.3.2. Persistent

Class ini adalah *base* class dari class-class lainnya yang ada di *package* ta.base.persistent.db. Inisialisasi dari class `Datastore`, `Session` dan `SessionFactory` dilakukan dalam class ini

```
public abstract class Persistent
{
      public Persistent(Class persistentClass) throws Exception {}

      private static Session openSession() throws HibernateException, SQLException
      {}

      public Serializable insert(Object object)
            throws HibernateException, SQLException {}

      public void update(Object object, Serializable id)
            throws HibernateException, SQLException {}

      public void delete(Object object) throws HibernateException, SQLException {}

      public Object load(Serializable id) throws HibernateException, SQLException {}

      public Session getSession() throws HibernateException, SQLException {}
      ...
}
```

Gambar 5.21 Fungsi-fungsi dalam class Persistent

Parameter berupa persistent class dari domain model diperlukan oleh konstruktor class. *Method* `getSession()` digunakan untuk mendapatkan `Session` *object*. `Session` *object* merupakan abstraksi class `java.sql.Connection` dari Hibernate. Sedangkan *method* `openSession()` digunakan untuk mendapatkan *object* `Session` dari `SessionFactory` (`SessionFactory.openSession()`). Gambar 5.22 merupakan detail implementasi dari *method* `getSession()` dan `openSession()`.

Class ini digunakan untuk melakukan operasi-operasi *create, update,* dan *delete* (CRUD). *Method* insert digunakan untuk menyimpan *persistent object* dari class. Berikut ini merupakan detail dari method `insert()`.

```
public Session getSession() throws HibernateException, SQLException
{
      if((!session.isConnected())||(!session.isOpen()))
            Persistent.session = openSession();

      return session;
}
// open session
private static Session openSession() throws HibernateException,
      SQLException
{
      Session session = sessionFactory.openSession();
      return session;
}
```

Gambar 5.22 Detail method getSession() dan openSession()

```
public Serializable insert(Object object)
            throws HibernateException, SQLException
{
      this.getSession();
      Serializable id = session.save(object);
      this.commit();
      return id;
}
```

Gambar 5.23 Detail method insert().

Berikut ini detail inisialisasi dari class `Persistent` untuk membuat *instance* dari `SessionFactory`.

```
static {
      try
      {
            PropertiesLoader loader = PropertiesLoader.instance();
            Properties properties = Environment.getProperties();

            String hbmFile =  properties.getProperty("hbmFile",
                              "ta.hbm.xml");
            ClassLoader classLoader =
                  Persistent.class.getClassLoader();
            InputStream inputStream =
                  classLoader.getResourceAsStream(hbmFile);
            // create data store
            datastore =
               Hibernate.createDatastore().
                  storeInputStream(inputStream);
            // build session factory
            sessionFactory = datastore.buildSessionFactory();
      } catch (Exception e)
      {
            e.printStackTrace();
      }
}
```

Gambar 5.24 Inisialisasi SessionFactory

Method `update()` digunakan untuk melakukan operasi update persistent object. Sedangkan *method* `delete()` digunakan untuk menghapus persistent object. *Method* `load()` digunakan untuk mengambil *persistent object* dari database.

Hal pertama yang dilakukan adalah membuat *instance* dari `Datastore`. File hbm.xml diperlukan class sebagai *mapping* antara *persistent class* dengan table pada database. `Datastore` mengelola *mapping* antara field-field didalam table dengan properti dari masing-masing class domain model. Selanjutnya `SessionFactory` dibuat dari *instance* dari `Datastore` tersebut melalui *method* `buildSessionFactory()`.

5.3.3. CClassPersistent

Class ini digunakan untuk melakukan *persistent activity* dari class `CClass`.

```
public class CClassPersistent extends Persistent {
    public static final String FULL_CLASS_NAME =
            Persistent.BASE_PERSISTENT_PACKAGE + "CClassModel";

    public CClassPersistent() throws Exception {}

    public long insert(String className, Set documents) throws
            Exception {}

    public void update(long classCode, String className, Set
           documentModels) throws Exception {}

    public void delete(long classCode) throws Exception {}

    public CClassModel load(long classCode) throws Exception {}

    public List findByClassName(String className) throws Exception
    {}
    ...
}
```

Gambar 5.25 Fungsi-fungsi class CClassPersistent

Method `insert()` digunakan untuk melakukan insert dari object class CClassModel. *Method* ini mengembalikan nilai berupa cclassCode yang merupakan kode dari persistent object tersebut didalam database. *Method* update() digunakan untuk melakukan update , dan delete digunakan untuk menghapus object dari table cclass.

BAB VI
UJI COBA DAN EVALUASI SISTEM

Pengujian pada perangkat lunak dilakukan setelah proses perancangan dan implementasi selesai. Uji coba dilakukan untuk mengetahui apakah dengan menggunakan *word cluster* dapat memecahkan permasalahan-permasalahan yang menjadi latar belakang dibuatnya aplikasi itu sendiri.

6.1. LINGKUNGAN UJI COBA

Lingkungan uji coba adalah komputer dimana uji coba sistem dan aplikasi dilakukan. Komputer yang digunakan untuk uji coba adalah komputer dengan prosessor Intel Pentium 4 1.7 GHz dengan kapasitas RAM sebesar 256 Mb, kapasitas hard disk 20 Gb. Untuk spesifikasi software-nya adalah operating system Windows XP Profesional, database MySQL sebagai tempat penyimpanan data.

6.2. DATA UJI COBA

Stopword yang digunakan adalah *stoplist* yang disusun oleh Gerard Salton dan Chris Buckley untuk aplikasi mereka *SMART Information Restrieval System* di Universitas Cornell. Stoplist ini terdiri dari 571 kata.

Sedangkan dataset yang digunakan dalam pengujian ini adalah dataset yang terdiri dari 19.997 artikel yang berasal dari 20 topik bahasan yang berasal dari Newsgroup (20NG). Dataset ini dikumpulkan oleh Ken Lang dan merupakan

salah satu dataset standard yang digunakan untuk menguji algoritma klasifikasi dokumen.

Dataset 20NG dapat dibagi berdasarkan kedekatan temanya menjadi 9 kelompok [Baker-2003] yaitu:

Table 6.1 Pembagian 20NG berdasarkan kedekatan topik

Groups	Content
Group 1	(a) talk.religion.misc; (b) soc.religion.christian; (c) alt.atheism
Group 2	(a) rec.sport.hockey; (b) rec.sport.baseball
Group 3	(a) talk.politics.mideast
Group 4	(a) sci.med; (b) talk.politics.guns; (c) talk.politics.misc
Group 5	(a) rec.autos; (b) rec.motorcycles; (c) sci.space
Group 6	(a) comp.os.ms-windows.misc; (b) comp.graphics; (c) comp.windows.x
Group 7	(a) sci.electronics; (b) comp.sys.mac.hardware; (c) comp.sys.ibm.pc.hardware
Group 8	(a) sci.crypt
Group 9	(a) misc.forsale

6.4. PELAKSANAAN UJI COBA

Uji coba dilaksanakan dalam beberapa skenario menguji keakurasian klasifikasi dengan menghilangkan *stopword* dan atau tanpa menggunakan *stemming*. Pada masing-masing skenario uji coba dilakukan pengujian klasifikasi dengan selisih fungsi obyektif untuk menghentikan iterasi pembuatan klaster sebesar 0.001.

6.4.1. Skenario Uji Coba

Pada bagian ini dilakukan serangkaian skenario uji coba. Skenario uji coba dilaksanakan untuk menguji klasifikasi. Dengan mengubah nilai beberapa parameter serta membandingkan hasil klasifikasi dengan menggunakan metode *feature selection*. Ada 3 skenario uji coba yaitu:

6.4.1.1 Uji Coba Pengaruh Jumlah Dokumen Training Untuk Membentuk Klaster

Uji coba ini digunakan untuk menguji pengaruh jumlah dokumen training yang digunakan untuk membentuk word cluster terhadap kemampuan klasifikasi. Pada uji coba ini tidak digunakan stemming. Masing-masing uji coba digunakan 3 *word cluster* dimana masing-masing *word cluster* diuji dengan menggunakan algoritma klasifikasi naïve bayes sebanyak 10 kali. Jumlah klaster dari setiap *word cluster* adalah 20 buah. Setiap klasifikasi menggunakan 100 dokumen penguji untuk setiap kategori. Uji coba ini terdiri dari beberapa bagian yaitu:

a. Uji Coba I

Menguji keakurasian penggunaan *word cluster* pada klasifikasi dengan metode naïve bayes dengan 20 dokumen *training* untuk tiap kategori.

b. Uji Coba II

Menguji keakurasian penggunaan *word cluster* pada klasifikasi dengan metode naïve bayes dengan 40 dokumen *training* untuk tiap kategori.

c. Uji Coba III

Menguji keakurasian penggunaan *word cluster* pada klasifikasi dengan metode naïve bayes dengan 70 dokumen *training* untuk tiap kategori.

d. Uji Coba IV

Menguji keakurasian penggunaan *word cluster* pada klasifikasi dengan metode naïve bayes dengan 100 dokumen *training* untuk tiap kategori

e. Uji Coba V

Menguji keakurasian penggunaan *word cluster* pada klasifikasi dengan metode naïve bayes dengan 200 dokumen *training* untuk tiap kategori.

6.4.1.2 Uji Coba Dengan Jumlah Word Cluster Yang Berbeda

Uji coba ini digunakan untuk menguji pengaruh jumlah word cluster yang terhadap kemampuan klasifikasi. Pada uji coba ini digunakan stemming, sedangkan jumlah dokumen 1/3 test 2/3 train split.. Uji coba dilakukan dengan menggunakan word cluster sebanyak 10, 20, 30, 80, 100, dan 150 buah.

6.4.1.3 Uji Coba Perbandingan Keakurasian Klasifikasi Menggunakan Word Clustering dengan Feature Selection

Uji coba ini digunakan untuk menguji keakurasian klasifikasi menggunakan word cluster sebagai *feature* dibandingkan dengan *feature selection*. *Feature selection* yang digunakan adalah *feature selection* yang menggunakan *mutual information* untuk *multinomial model* [McCal-1998].

Ada dua skenario yang digunakan pada uji coba ini dimana skenario pertama digunakan stemming dan dokumen training yang dibagi dalam 1/3 test 2/3 train split dari data set yang digunakan. Uji coba dilakukan dengan menggunakan *feature* sebanyak 10, 20, 30, 80, 100, dan 150 dari masing-masing *feature selection* maupun *feature clustering*.

Pada skenario kedua juga digunakan stemming dan dokumen training yang kecil yaitu 150 dokumen training untuk setiap class label. Uji coba dilakukan dengan jumlah *feature* 10, 15, 20, 25, 40 buah untuk masing-masing *feature selection* maupun *feature word clustering*.

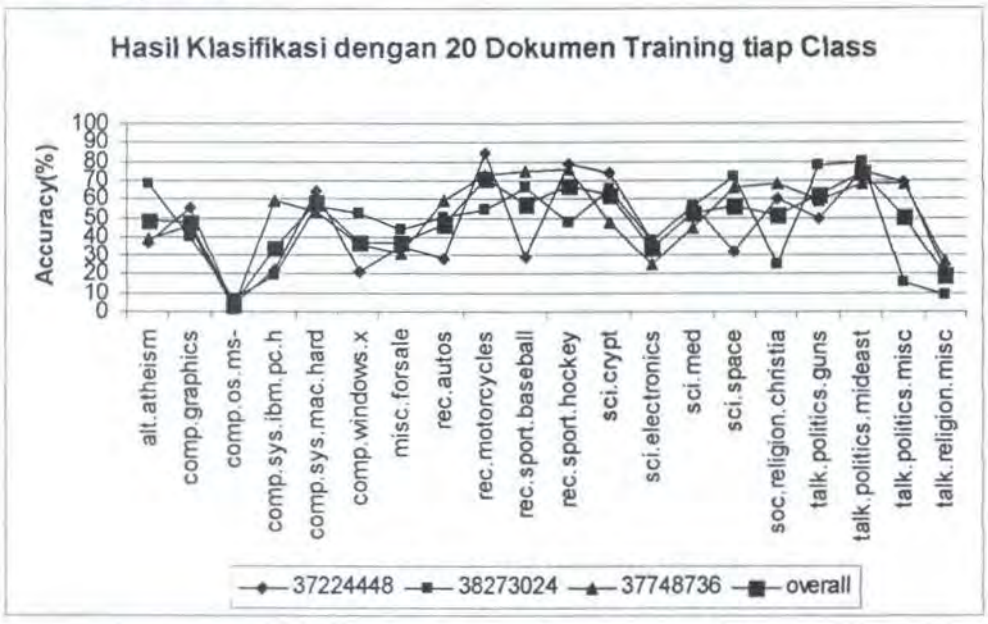
Pada setiap skenario uji coba, dokumen yang digunakan untuk menguji sebesar 100 untuk masing-masing class label, sehingga total dokumen yang digunakan untuk menguji adalah 2000 buah.

6.4.2. Hasil Pelaksanaan Uji Coba

6.4.2.1 Hasil Klasifikasi dengan Jumlah Dokumen Training yang berbeda.

Berikut ini adalah gambar grafik hasil klasifikasi dari uji coba yang dilaksanakan.

a. Hasil Uji Coba I



Gambar 6.1 Hasil Klasifikasi dengan 20 Dokumen Training tiap Class

Word cluster dibentuk dari 20 buah dokumen training untuk masing-masing class label masing class label (total dokumen training 20 x 20 = 400

dokumen) atau hanya 2% dari total keseluruhan dokumen 20NG yang berjumlah 20000 dokumen. Dari klasifikasi dengan *word cluster* ini diperoleh hasil yang bervariasi antara 10%-70%.

b. Hasil Uji Coba II

Gambar 6.2 Hasil Klasifikasi dengan 40 Dokumen Training tiap Class

Word cluster dibentuk dari 40 buah dokumen training untuk masing-masing class label (total dokumen training 40 x 20 = 800 dokumen) atau hanya 4% dari total keseluruhan dokumen 20NG yang berjumlah 20000 dokumen. Secara keseluruhan klasifikasi ini pada uji coba II lebih baik dari pada hasil yang diperoleh pada uji coba I

c. Hasil Uji Coba III

Word cluster dibentuk dari 70 buah dokumen training untuk masing-masing class label (total dokumen training 70 x 20 = 1400 dokumen) atau hanya 7% dari total keseluruhan dokumen 20NG yang berjumlah 20000 dokumen.

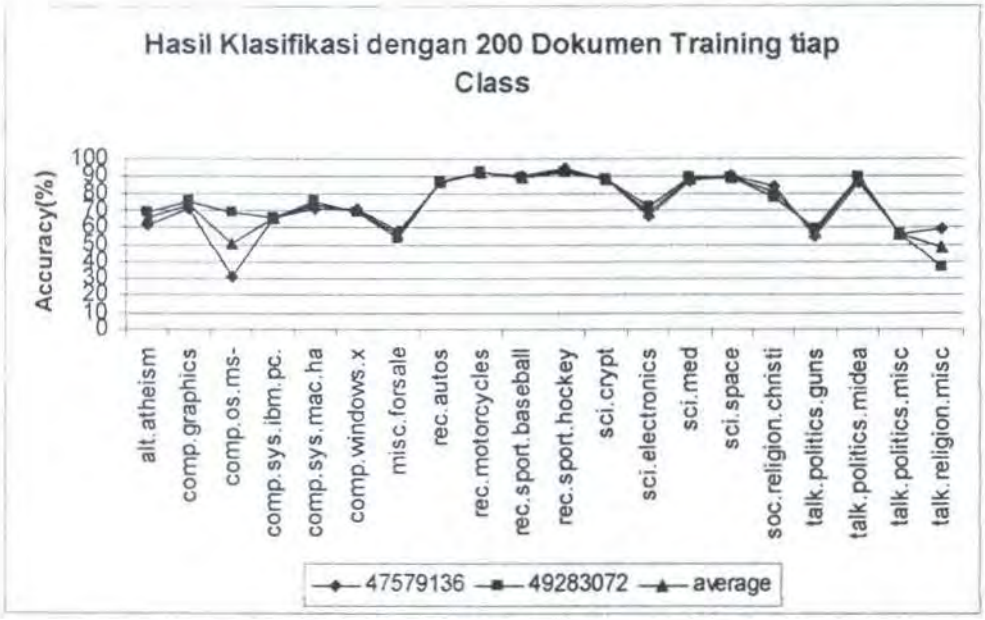
Gambar 6.4 Hasil Klasifikasi dengan 70 Dokumen Training Tiap Class

d. Hasil Uji Coba IV

Gambar 6.5 Hasil Klasifikasi dengan 100 Dokumen Training Tiap Class

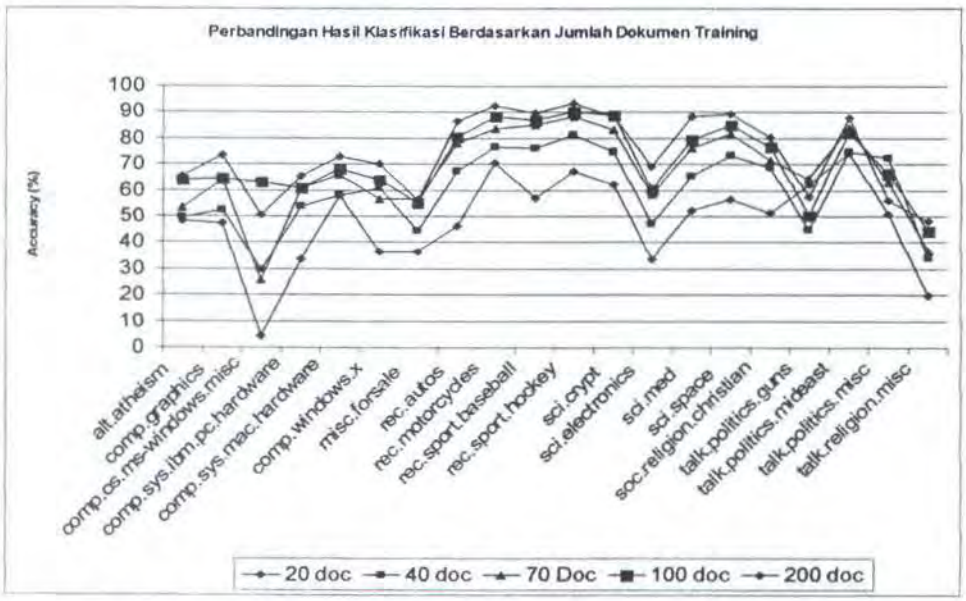
Word cluster dibentuk dari 100 buah dokumen training untuk masing-masing class label (total dokumen training 100 x 20 = 2000 dokumen) atau hanya 10% dari total keseluruhan dokumen 20NG yang berjumlah 20000 dokumen. Pada uji coba ini diperoleh hasil yang bervariasi antara 40%-95%.

e. Hasil Uji Coba V



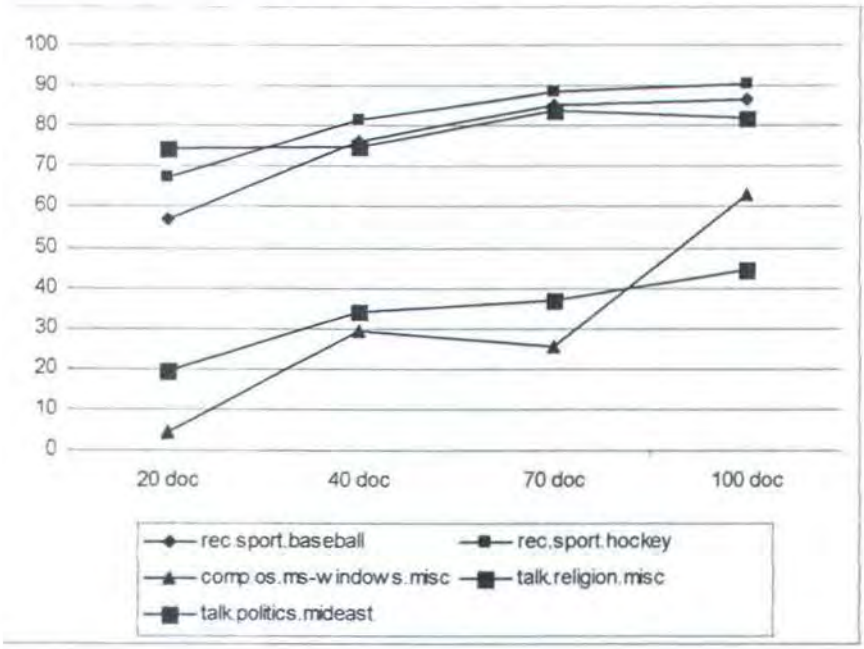
Gambar 6.6 Hasil Klasifikasi dengan 200 dokumen Training tiap class

Word cluster dibentuk dari 200 buah dokumen training untuk masing-masing class label (total dokumen training 100 x 20 = 2000 dokumen) atau hanya 20% dari total keseluruhan dokumen 20NG yang berjumlah 20000 dokumen.



Gambar 6.7 Perbandingan Hasil Klasifikasi berdasarkan jumlah dokumen training

Pada uji coba ini diperoleh hasil batas bawah yang lebih rendah dari pada uji coba IV yaitu bervariasi antara 30%-95%.

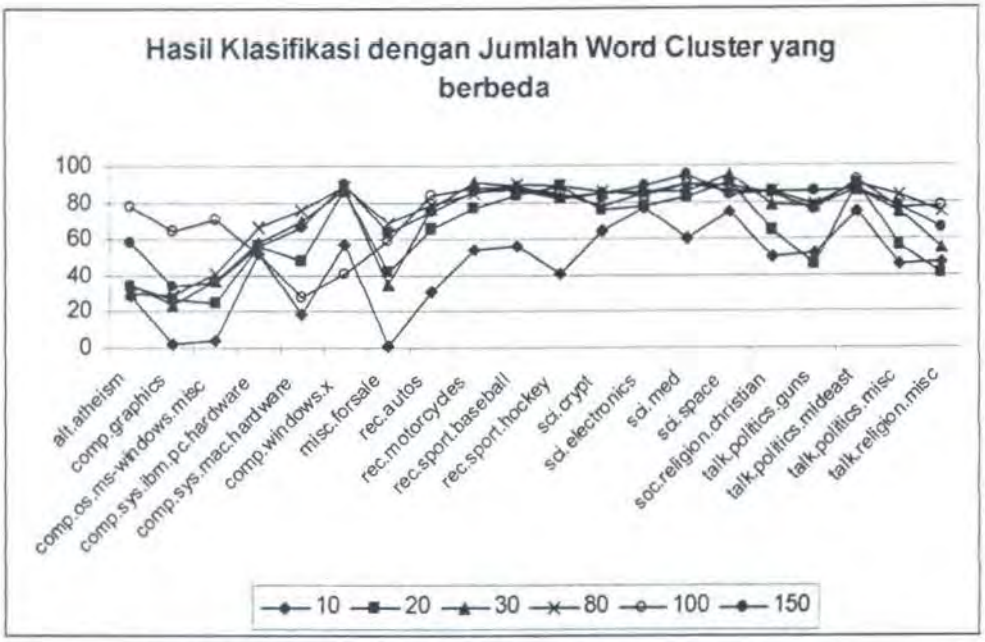


Gambar 6.8 Pengaruh jumlah dokumen training dalam membentuk klaster terhadap keakurasian klasifikasi

Pada gambar 6.7 dan gambar 6.8 dapat diketahui bahwa dengan menggunakan jumlah dokumen training yang semakin banyak maka keakurasian hasil klasifikasi akan semakin baik.

6.4.2.2 Hasil Klasifikasi Dengan Jumlah Word Cluster Yang Berbeda

Pada gambar 6.9 ditunjukan hasil klasifikasi dengan menggunakan jumlah word cluster yang berbeda pada dataset 1/3 test 2/3 train split. Dari gambar tersebut dapat dilihat bahwa dengan menggunakan *word cluster* sebesar 10 buah diperoleh hasil klasifikasi yang rendah. Dengan menggunakan *feature* berupa *word cluster* sebesar 20, 30, 80, 100, 150 diperoleh hasil yang hampir sama pada beberapa class label. Hasil klasifikasi yang palingbaik diperoleh dengan menggunakan word cluster sebanyak 100.

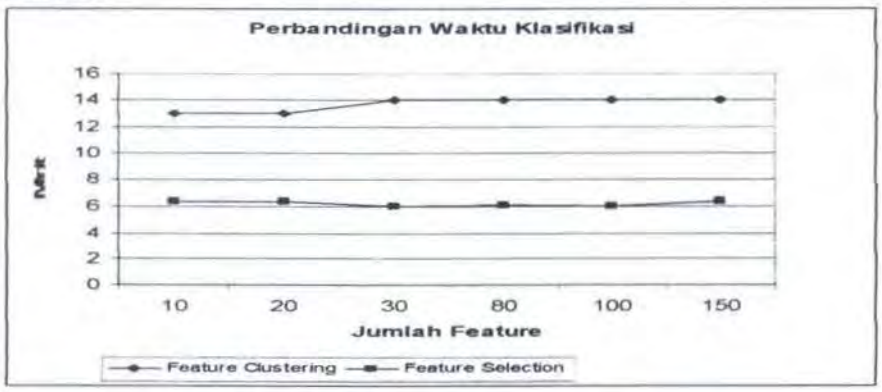


Gambar 6.9 Hasil klasifikasi dengan jumlah word cluster yang berbeda

Secara keseluruhan dapat diperoleh kesimpulan bahwa untuk hasil klasifikasi yang baik jumlah *word cluster* yang digunakan minimal harus sama dengan jumlah class label yang dipakai. Sedangkan pada hasil uji coba menunjukkan bahwa semakin besar jumlah *feature* yang digunakan akan meningkatkan waktu klasifikasi.

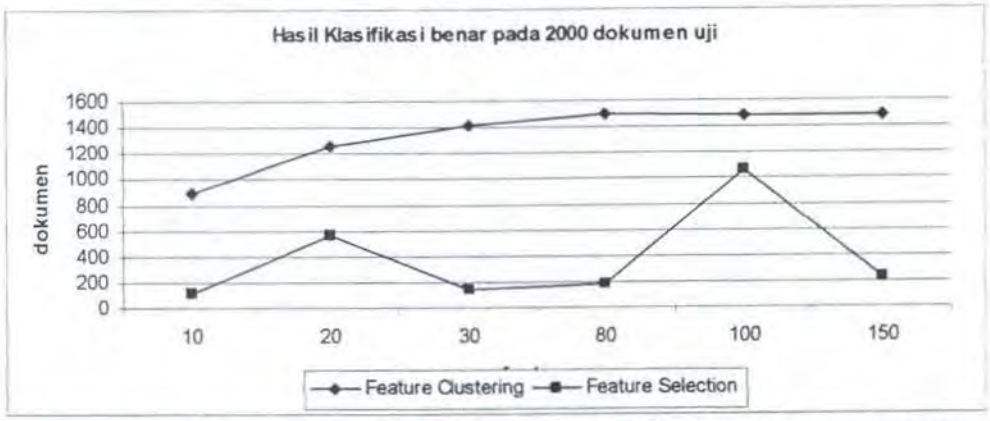
6.4.2.3 Perbandingan Feature Clustering dengan Feature Selection

- Skenario I



Gambar 6.10 Perbandingan Waktu klasifikasi pada 2000 dokumen uji antara Feature Selection dan Feature Clustering.

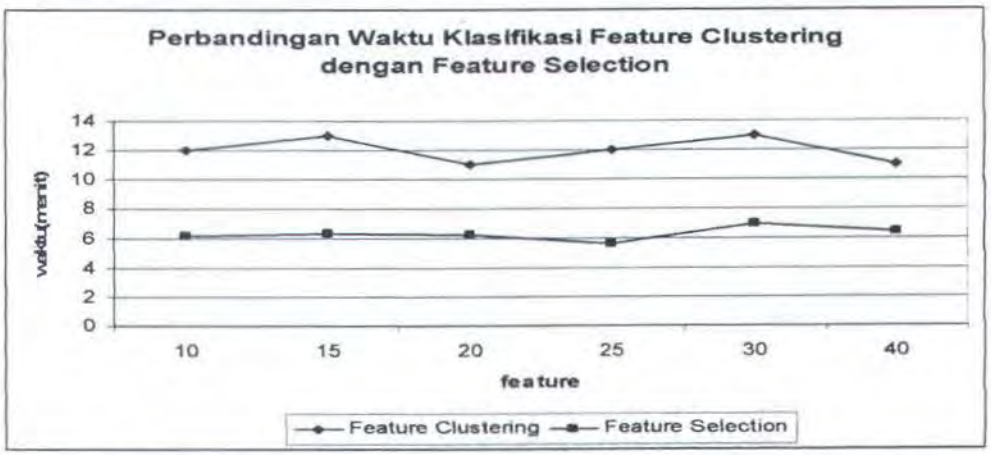
Pada gambar 6.10 ditunjukan perbandingan waktu klasifikasi skenario pertama antara *feature selection* dan *feature clustering.*



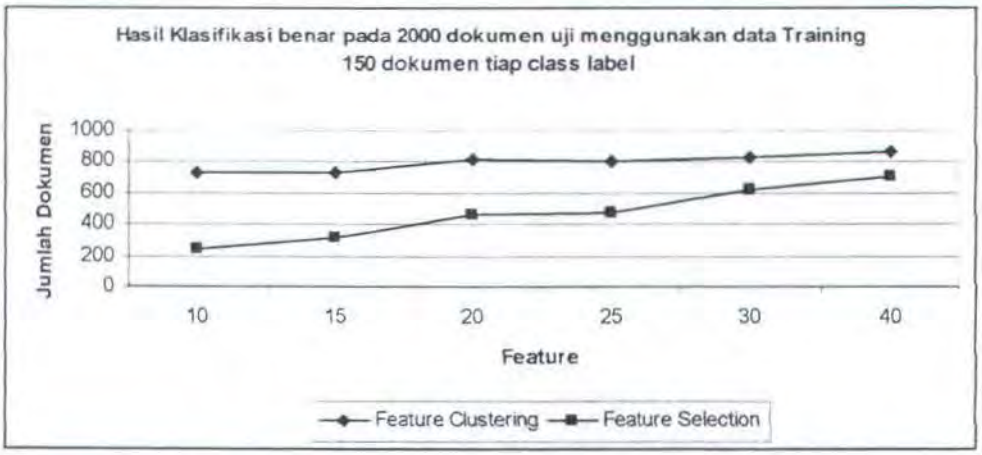
Gambar 6.11 Perbandingan hasil klasifikasi dengan berbagai jumlah feature antara Feature Selection dan Feature Clustering dengan 2000 dokumen setiap pengujian

Gambar 6.11 menunjukan juga hasil klasifikasi pada skenario pertama. Pada gambar tersebut dapat diketahui bahwa kemampuan klasifikasi dengan jumlah *feature* yang sedikit, *feature clustering* jauh lebih baik pada klasifikasi yang menggunakan jumlah *feature selection.* Sedangkan waktu yang digunakan untuk melakukan klasifikasi dengan jumlah *feature* sedikit *feature selection* jauh lebih unggul dibandingkan dengan *feature clustering.*

- Skenario II

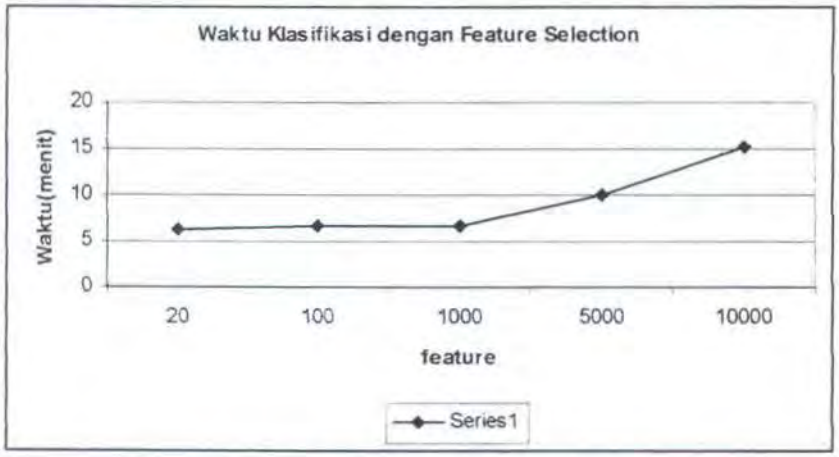


Gambar 6.12 perbandingan waktu klasifikasi antara feature clustering dengan feature selection



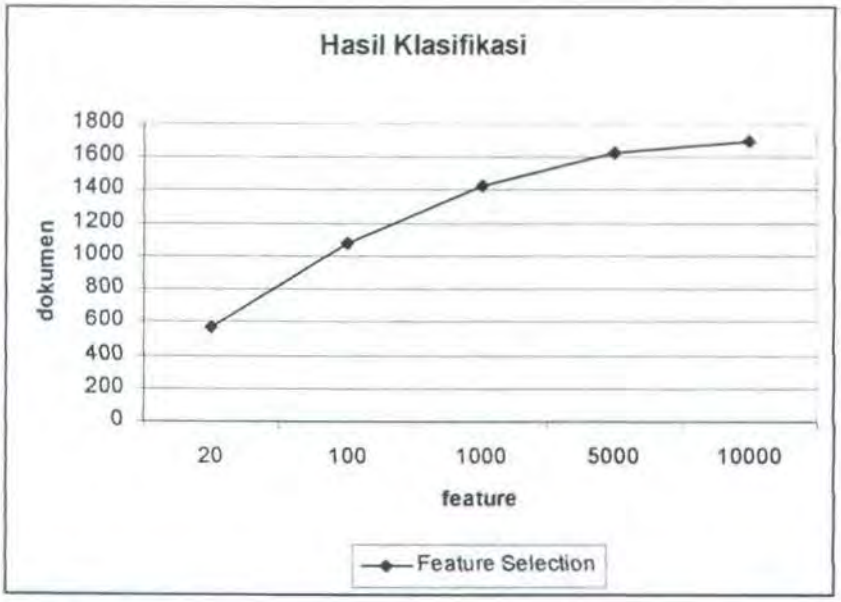
Gambar 6.13. Perbandingan hasil klasifikasi antara feature clustering dengan feature selection

Pada Gambar 6.12 ditunjukkan waktu klasifikasi skenario kedua antara *feature selection* dengan *feature clustering*. Sedangkan pada gambar 6.13 ditunjukkan perbandingan hasil dokumen yang terklasifikasi benar antara *feature selection* dengan *feature clustering*.



Gambar 6.13 waktu komputasi feature selection

Dari hasil kedua skenario tersebut dapat diambil beberapa kesimpulan antara lain bahwa hasil klasifikasi dengan menggunakan *feature clustering* memiliki keakurasian klasifikasi yang lebih baik baik dibandingkan dengan *feature selection* pada jumlah *feature* yang sedikit, akan tetapi waktu komputasi lebih besar dari pada *feature selection*. dengan menggunakan *feature* yang sedikit.



Gambar 6.14. hasil klasifikasi dengan menggunakan feature selection

Gambar 6.13 menunjukkan waktu komputasi yang diperlukan oleh naïve bayes dengan *feature selection*. Pada gambar tersebut menunjukkan waktu komputasi akan meningkat jika jumlah *feature* yang digunakan semakin besar. akan tetapi tingginya waktu komputasi dengan *feature selection* diimbangi dengan semakin baiknya hasil klasifikasi.

6.5. EVALUASI

Dari hasil uji coba yang telah dilakukan, maka dapat dievaluasi sebagai berikut:

a. Jumlah dokumen *training* yang digunakan mempengaruhi tingkat keakurasian klasifikasi. Hal ini dapat dilihat pada tabel dibawah ini :

Table 6.2. Hasil klasifikasi dengan jumlah dokumen *training* yang berbeda

class	Jumlah Dokumen Training				
	20 doc	40 doc	70 doc	100 doc	200
alt.atheism	48.2	49.5	53.26666667	63.8	65.3
comp.graphics	47.1	52.16666667	64.03333333	64.4	73.3
comp.os.ms-windows.misc	4.233333333	29.46666667	25.5	63.2	50.25

comp.sys.ibm.pc.hardware	33.53333333	53.66666667	60.93333333	60.76667	65.55
comp.sys.mac.hardware	58.16666667	57.73333333	65.7	68.26667	72.95
comp.windows.x	36.56666667	60.9	56.4	63.9	69.9
misc.forsale	36.56666667	44.5	56.76666667	54.93333	56.1
rec.autos	46.13333333	67.3	78	80.36667	86.6
rec.motorcycles	70.73333333	76.56666667	83.53333333	88.26667	92.65
rec.sport.baseball	56.8	76.3	85.26666667	86.93333	89.65
rec.sport.hockey	67.43333333	81.4	88.43333333	90.36667	93.45
sci.crypt	61.93333333	75	83.26666667	88.56667	88.2
sci.electronics	33.66666667	47.03333333	58.63333333	60.43333	69.35
sci.med	52.3	65.2	76.16666667	79.4	88.45
sci.space	56.6	73.4	81.73333333	84.9	89.4
soc.religion.christian	51.4	68.56666667	71.46666667	76.5	80.4
talk.politics.guns	62.03333333	44.76666667	64.4	50.46667	57.45
talk.politics.mideast	74.26666667	74.86666667	84	82.16667	87.85
talk.politics.misc	50.86666667	72.4	63.13333333	66.53333	55.9
talk.religion.misc	19.56666667	33.93333333	37.06666667	44.43333	48.05

Dari tabel diatas dapat kita ketahui bahwa semakin besar jumlah dokumen training yang digunakan tingkat keakurasian klasifikasi semakain baik. Pada class comp.os.ms-windows.misc dengan menggunakan 20 dokumen diperoleh hasil klasifikasi dibawah 10% sedangkan dengan menggunakan 100 diperoleh hasil klasifikasi 60%.

b. Untuk mendapatkan hasil klasifikasi yang baik dengan menggunakan *word cluster* maka jumlah *word cluster* yang digunakan minimal sama dengan jumlah class label.

Tabel 6.3 Hasil Klasifikasi dengan *word cluster* yang berbeda

class	feature size					
	10	20	30	80	100	150
alt.atheism	29	34	35	30	78	58
comp.graphics	2	27	23	29	64	34
comp.os.ms-windows.misc	4	24	37	41	70	36
comp.sys.ibm.pc.hardware	52	56	58	66	52	56
comp.sys.mac.hardware	18	48	69	76	28	66
comp.windows.x	57	87	87	89	41	90
misc.forsale	1	42	35	68	59	63
rec.autos	31	65	77	79	84	75
rec.motorcycles	54	77	91	85	88	86
rec.sport.baseball	56	84	89	90	88	87
rec.sport.hockey	41	89	83	89	85	82
sci.crypt	64	76	85	86	78	85
sci.electronics	77	78	86	84	85	89

sci.med	60	83	86	90	90	95
sci.space	75	91	95	86	89	84
soc.religion.christian	50	64	79	86	85	86
talk.politics.guns	52	46	78	79	76	86
talk.politics.mideast	75	90	87	90	92	87
talk.politics.misc	46	56	74	84	77	78
talk.religion.misc	47	41	55	75	78	65

c. Hasil klasifikasi dengan menggunakan *feature clustering* lebih baik dari pada *feature selection* pada jumlah *feature* yang sedikit (jumlah *feature* minimal sama dengan jumlah class label seperti yang disebutkan pada point b). seperti yang ditunjukan pada tabel 6.4.

Tabel 6.4 Perbandingan hasil klasifikasi antara *feature clustering* dengan *feature selection*

no	Jumlah document training per class	Feature Clustering			Feature Selection		
		Jumlah cluster	klasifikasi benar	waktu (menit)	Jumlah Feature	klasifikasi benar	waktu (menit)
1	150 doc train	10	730	12	10	240	6.15
2	150 doc train	15	731	13	15	323	6.32
3	150 doc train	20	814	11	20	464	6.19
4	150 doc train	25	804	12	25	475	5.56
5	150 doc train	30	834	13	30	616	7
6	150 doc train	40	865	11	40	706	6.37

BAB VII
KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan yang diambil dari hasil ujicoba yang telah dilaksanakan. Selanjutnya, diberikan beberapa saran yang mungkin dapat digunakan untuk mengembangkan hasil yang diperoleh pada tugas akhir ini.

7.1. KESIMPULAN

Kesimpulan yang diambil dari hasil uji coba yang telah dilakukan adalah:

1. Penggunaaan *word cluster* sebagai penganti *feature* dalam klasifikasi dokumen merupakan alternatif yang bisa digunakan untuk mengurangi tingginya dimensi *feature space* yang menjadi masalah pada komputasi algoritma klasifikasi selain dengan menggunakan *feature selection*.
2. Hasil klasifikasi dengan menggunakan *feature clustering* lebih baik dari pada *feature selection* pada jumlah *feature* yang sedikit. Akan tetapi waktu komputasi menjadi lebih lama bila dibandingkan dengan *feature selection*. Jika dokumen training yang dimiliki sedikit, maka dengan menggunakan *feature clustering* diperoleh hasil yang lebih baik.
3. Jumlah *word cluster* minimal sama dengan jumlah class label untuk mendapatkan hasil klasifikasi yang baik.

7.2. SARAN

Saran-saran yang dapat diberikan untuk pengembangan tugas akhir ini adalah:

1. Perlu dilakukannya pembuatan pengklasifikasian dokumen dengan menggunakan algoritma lain selain naïve bayes yang memberikan hasil klasifikasi yang lebih baik.
2. Perlunya penggunaan algoritma *feature clustering* lain yang bisa memberikan hasil yang lebih baik.
3. Perlunya penanganan untuk dokumen yang berformat selain teks.

DAFTAR PUSTAKA

[Baker-1998] Baker, LD & Mc Callum, A, *"Distributional Clustering of Word for Text Classification"*. 21 Annual Internation ACM SIGIR, 1998.

[Beker-2003] Bekerman, Ron, dkk, *"Distributional Word Clusters vs Word for Text Classification"*, Journal of Machine Learning Research 3, 2003.

[Dhill-2003] Dhillon, Inderjit S, dkk, *"A Divisive Information-Theoretic Feature Clustering Algorithm for Text Classification"*, Jurnal of Machine Learning Research 3 (1265-1287), 2003.

[Jain-1999] Jain, Anil K, dkk, *"Statistical Pattern Recognition: a Review"*, IEEE, 1999

[King-2005] King, Gavin & Bauer, C, *"Hibernate in Action"*, Manning Publication, 2005.

[Lin-1991] Joushua, Lin, *"Divergence Measurement Based on the Shannon Entropy"*. IEEE Trans. Inform. Theory 37(1):145-151, 1991.

[McCal-1998] Mc Callum, A & Nigam, K, *"A Comparison of Event Model for Naïve Bayes Text Classification"*. AAAI-98 Workshop on Learning for Text Categorization, 1998.

[McKay-2003] Mc Kay, David, *"Information Theory, Inference, and Learning Algorithm"*. Cambridge University Press, 2001.

[Paice-1990] Paice, Chris D, *"Lancaster Paice/Husk Stemming Algorithm Article"*. Dept. of Computing Lancaster University, http://www.lancs.ac.uk/ug/oneillc1/stemmer/index.htm.

[Yang-1997] Yang, Yimin & Pedersen, J.O, *"A Comparative Study on Feature Selection in Text Classification"*, Proc. 14th International Conference on Machine Learning, pages 412-420, Morgan Kaufmann, 1997

LAMPIRAN

HASIL KLASIFIKASI DENGAN WORD CLUSTER YANG DIBENTUK DARI DOKUMEN TRAINING YANG BERBEDA

class	Document Training Size				
	20 doc	40 doc	70 doc	100 doc	200
alt.atheism	48.2	49.5	53.26666667	63.8	65.3
comp.graphics	47.1	52.16666667	64.03333333	64.4	73.3
comp.os.ms-windows.misc	4.233333333	29.46666667	25.5	63.2	50.25
comp.sys.ibm.pc.hardware	33.53333333	53.66666667	60.93333333	60.76667	65.55
comp.sys.mac.hardware	58.16666667	57.73333333	65.7	68.26667	72.95
comp.windows.x	36.56666667	60.9	56.4	63.9	69.9
misc.forsale	36.56666667	44.5	56.76666667	54.93333	56.1
rec.autos	46.13333333	67.3	78	80.36667	86.6
rec.motorcycles	70.73333333	76.56666667	83.53333333	88.26667	92.65
rec.sport.baseball	56.8	76.3	85.26666667	86.93333	89.65
rec.sport.hockey	67.43333333	81.4	88.43333333	90.36667	93.45
sci.crypt	61.93333333	75	83.26666667	88.56667	88.2
sci.electronics	33.66666667	47.03333333	58.63333333	60.43333	69.35
sci.med	52.3	65.2	76.16666667	79.4	88.45
sci.space	56.6	73.4	81.73333333	84.9	89.4
soc.religion.christian	51.4	68.56666667	71.46666667	76.5	80.4
talk.politics.guns	62.03333333	44.76666667	64.4	50.46667	57.45
talk.politics.mideast	74.26666667	74.86666667	84	82.16667	87.85
talk.politics.misc	50.86666667	72.4	63.13333333	66.53333	55.9
talk.religion.misc	19.56666667	33.93333333	37.06666667	44.43333	48.05

HASIL KLASIFIKASI DENGAN MENGGUNAKAN FEATURE CLUSTERING 1/3TEST2/3TRAIN SPLIT

class	feature size					
	10	20	30	80	100	150
alt.atheism	29	34	35	30	78	58
comp.graphics	2	27	23	29	64	34
comp.os.ms-windows.misc	4	24	37	41	70	36
comp.sys.ibm.pc.hardware	52	56	58	66	52	56
comp.sys.mac.hardware	18	48	69	76	28	66
comp.windows.x	57	87	87	89	41	90
misc.forsale	1	42	35	68	59	63
rec.autos	31	65	77	79	84	75

rec.motorcycles	54	77	91	85	88	86
rec.sport.baseball	56	84	89	90	88	87
rec.sport.hockey	41	89	83	89	85	82
sci.crypt	64	76	85	86	78	85
sci.electronics	77	78	86	84	85	89
sci.med	60	83	86	90	90	95
sci.space	75	91	95	86	89	84
soc.religion.christian	50	64	79	86	85	86
talk.politics.guns	52	46	78	79	76	86
talk.politics.mideast	75	90	87	90	92	87
talk.politics.misc	46	56	74	84	77	78
talk.religion.misc	47	41	55	75	78	65

HASIL KLASIFIKASI DENGAN MENGGUNAKAN FEATURE

SELECTION 1/3TEST2/3TRAIN SPLIT

class	Feature					
	10	20	30	80	100	150
alt.atheism	100	16	95	85	78	78
comp.graphics	1	3	0	2	61	9
comp.os.ms-windows.misc	0	3	0	4	55	2
comp.sys.ibm.pc.hardware	0	28	0	4	3	1
comp.sys.mac.hardware	0	10	0	1	19	0
comp.windows.x	0	61	1	8	5	7
misc.forsale	8	1	32	37	48	44
rec.autos	2	58	3	5	70	8
rec.motorcycles	0	48	2	3	79	7
rec.sport.baseball	0	48	0	2	59	7
rec.sport.hockey	0	38	0	9	66	9
sci.crypt	1	64	1	2	85	21
sci.electronics	0	3	0	2	21	3
sci.med	0	36	0	3	65	5
sci.space	1	1	0	5	63	1
soc.religion.christian	0	16	0	1	76	5
talk.politics.guns	0	38	0	0	53	3
talk.politics.mideast	0	13	1	8	81	17
talk.politics.misc	0	30	1	2	40	5
talk.religion.misc	0	49	0	1	34	0

PERBANDINGAN KEAKURASIAN & WAKTU KLASIFIKASI FEATURE SELECTION DAN FEATURE CLUSTERING 1/3TEST 2/3TRAIN SPLIT

no	Document Training Size per class	Feature Clustering			Feature Selection		
		Jumlah cluster	klasifikasi benar	waktu (menit)	Jumlah Feature	klasifikasi benar	waktu (menit)
1	1/3test2/3traini split	10	891	13	10	113	6.39
2	1/3test2/3traini split	20	1258	13	20	564	6.39
3	1/3test2/3traini split	30	1409	14	30	136	6
4	1/3test2/3traini split	80	1502	14	80	184	6.09
5	1/3test2/3traini split	100	1487	14	100	1061	5.99
6	1/3test2/3traini split	150	1488	14	150	232	6.37

HASIL KLASIFIKASI DENGAN MENGGUNAKAN FEATURE CLUSTERING 150 DOKUMEN TRAINING PER CLASS LABEL

class	feature size					
	10	15	20	25	30	40
alt.atheism	16	19	20	11	23	17
comp.graphics	0	9	11	12	13	14
comp.os.ms-windows.misc	21	1	14	14	7	20
comp.sys.ibm.pc.hardware	22	3	16	16	31	15
comp.sys.mac.hardware	9	20	30	39	26	26
comp.windows.x	39	29	35	31	36	35
misc.forsale	3	10	13	20	28	23
rec.autos	26	31	26	28	25	37
rec.motorcycles	44	39	38	42	37	44
rec.sport.baseball	60	53	48	54	43	45
rec.sport.hockey	62	62	19	54	51	52
sci.crypt	30	37	54	54	44	53
sci.electronics	60	47	53	56	61	47
sci.med	67	50	52	52	68	46
sci.space	39	63	61	48	65	68
soc.religion.christian	22	45	57	41	31	56
talk.politics.guns	56	32	60	47	43	63
talk.politics.mideast	39	49	72	56	58	66
talk.politics.misc	63	73	77	81	89	63
talk.religion.misc	52	59	58	48	55	75

HASIL KLASIFIKASI DENGAN MENGGUNAKAN FEATURE SELECTION 150 DOKUMEN PER CLASS LABEL

class	Feature					
	10	15	20	25	30	40
alt.atheism	5	8	7	4	6	9
comp.graphics	4	1	1	19	8	12
comp.os.ms-windows.misc	1	30	1	12	1	24
comp.sys.ibm.pc.hardware	8	0	10	11	31	23
comp.sys.mac.hardware	24	0	9	12	12	12
comp.windows.x	24	27	32	24	42	42
misc.forsale	24	1	3	9	1	2
rec.autos	9	11	14	57	57	61
rec.motorcycles	3	24	62	9	31	64
rec.sport.baseball	9	10	34	32	33	25
rec.sport.hockey	29	34	38	53	49	73
sci.crypt	43	64	57	54	76	67
sci.electronics	6	15	10	8	8	17
sci.med	18	30	33	47	42	63
sci.space	18	4	13	10	28	24
soc.religion.christian	0	6	40	17	31	33
talk.politics.guns	9	18	36	12	43	32
talk.politics.mideast	3	11	30	23	51	53
talk.politics.misc	2	8	24	27	29	35
talk.religion.misc	1	21	10	35	37	35

PERBANDINGAN KEAKURASIAN & WAKTU KLASIFIKASI FEATURE SELECTION DAN FEATURE CLUSTERING 150 DOKUMEN TRAINING PER CLASS LABEL

no	Document Training Size per class	Feature Clustering			Feature Selection		
		Jumlah cluster	klasifikasi benar	waktu (menit)	Jumlah Feature	klasifikasi benar	waktu (menit)
1	150 doc train	10	730	12	10	240	6.15
2	150 doc train	15	731	13	15	323	6.32
3	150 doc train	20	814	11	20	464	6.19
4	150 doc train	25	804	12	25	475	5.56
5	150 doc train	30	834	13	30	616	7
6	150 doc train	40	865	11	40	706	6.37