Jurnal Ilmiah KURSOR Menuju Solusi Teknologi Informasi
Vol. 4, No. 2, 2008
ISSN 0216 - 0544

PENGKLASIFIKASIAN *WEB PAGE* MELALUI PENDEKATAN *DECISION TREE*

Victor Hariadi
Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Gedung Teknik Informatika, Kampus ITS, Jl. Raya ITS, Sukolilo, Surabaya – 60111
email : victor@its-sby.edu

Abstract

Document/text classifications is text processing and labelling to any documents containing text. Text is natural languange which can understanding by human. In recent 1980, the approach that used to text/document classification is by manual construction which built by knowledge engineering technique. An expert system to support decision making by the text classifications. But this approach has an fundamentaly weaknesses. We had many limitations of information became from expert system literatures. If we want to updating the categories set, so we have to start the text classification from the beginning.

In our research, we used decision tree method combined with Oracle Text 9.2 text searching features to build new and robust document/text classification system.

Keywords: *document/text classification, categories set, decision tree*

1. Pendahuluan

Meski terkesan sepele, namun sebenarnya pengklasifikasian dokumen menjadi hal yang sangat penting. Seringkali para pengguna internet menggunakan berbagai macam teknik dan *keyword-keyword* unik untuk menemukan halaman *web* yang dicari. Banyak pengguna intenet mengeluhkan besarnya waktu yang digunakan untuk mencari halaman spesifik yang diinginkan. Karena sangat mungkin *search engine* menghasilkan *query* halaman *web* yang mengandung *keyword* yang dimaksud namun secara keseluruhan halaman *web* itu bukanlah berisi tentang informasi yang dimaksud oleh *keyword* itu melainkan hanya menyebutkan kata itu saja, misalnya *blog* pribadi seseorang yang menyebut sistem ekonomi Indonesia, padahal halaman itu bukan merupakan halaman yang membahas materi yang dimaksud pencarinya.

Pengklasifikasian mungkin dapat dilakukan secara manual, misalnya ada seorang yang bertugas membaca dan menyimpulkan halaman-halaman dengan satu atau dua kata kunci. Misalnya "pendidikan", "hiburan", "politik" dan lain-lain. Namun, seiring dengan berkembangnya teknologi informasi dan melekatnya konsep paperless dalam kehidupan sehari-hari telah menyebabkan permasalahan sepele ini menjadi masalah yang cukup serius. Karena tentu saja pengklasifikasian dokumen akan membutuhkan waktu yang sangat lama terlebih jika ternyata dilakukan secara manual dan dokumen-dokumen itu memang membahas lebih dari satu kelas seperti "ekonomi" dan "politik".

2. Klasifikasi Dokumen

Information retrieval adalah salah satu disiplin ilmu yang mempelajari cara menemukan kembali informasi dengan melakukan penelusuran kumpulan informasi yang telah diperoleh sebelumnya. Dengan IR ini, pengklasifikasian dokumen dapat ditelaah lebih baik.

Kinerja sistem temu kembali informasi terdiri atas 3 elemen dasar, yaitu : *input*, proses, dan *output*. Pada sisi *input*, permasalahannya adalah cara menentukan representasi dari dokumen yang mampu diproses (dalam bentuk bentuk *query*) pada sebuah komputer. Dalam sistem ini juga lebih dipilih menggunakan bahasa *artificial* daripada bahasa *natural* dalam menjalankan proses *query* dan memformulasikan dokumen.

Pada bagian proses, biasanya terdiri atas proses strukturisasi informasi (seperti pengklasifikasian) atau melakukan eksekusi strategi pencarian sebuah respon dari adanya *query*. Dokumen yang menjadi masukan dapat dilibatkan kembali selama tahap proses untuk menghasilkan *output*.

Teknik IR banyak digunakan dalam tiga fase pengklasifikasian teks, terdiri atas [14]:

- Tahap *preprocessing* termasuk proses *indexing*
- Pengklasifikasian melalui proses pembelajaran (dalam sistem ini menggunakan *decision tree classifier*)
- Evaluasi untuk kerja pengklasifikasi.

3. *Prepocessing*

Tahap ini merupakan tahap saat dokumen tekstual ditangani dan disiapkan untuk membuat dokumen dapat diproses oleh *decision tree classifier*.

Ada 5 tahap prosedur teks *preprocessing* [1] :

3.1. Analisa Leksikal

Analisa leksikal (*lexical analyzer*) adalah suatu proses transformasi teks yang masuk menjadi kata-kata, yang nantinya akan diadaptasikan sebagai indeks kata. Pada tahap ini dilakukan pengidentifikasian dan penanganan spasi sebagai pemisah kata, digit, tanda hubung, tanda baca, besar kecil huruf yang harus diperhatikan, serta nomer, yang biasanya tidak dianggap dalam indeks kata. Penetapan analisa leksikal dilakukan sesuai *preference* Lexer di *Oracle Text* 9.2.

3.2. Penghilangan *Stopword*

Dalam suatu kalimat bahasa Inggris, sering muncul beberapa kata yang tidak dapat digunakan sebagai *term* pengindeks, seperti 'the', 'of','and','to', dan lain-lain. Kata-kata yang sering muncul ini disebut sebagai *stopword* dan biasanya dihilangkan dari *list* indeks kata. Preposisi kata hubung dan partikel biasanya merupakan kandidat *list stopword* yang disebut *stoplist*, yaitu suatu daftar kata yang dapat dibuang dalam proses pengindeksan, karena mengurangi kualitas term pengindeks [4].

Untuk melakukan penyaringan kata-kata *stoplist* dalam aliran pertama dapat dilakukan dengan beberapa cara. Pendekatan yang dianggap terbaik adalah dengan membuang *stopword* sebagai bagian dari analisa leksikal, karena dengan mengikutsertakan pembuangan *stopword* pada proses analisa leksikal, pengenalan *stopword* dari daftar yang berukuran besar dapat dilakukan dengan *cost* lebih kecil dan lebih mudah. *Stoplist* yang berisi kumpulan *stopword* otomatis dihilangkan dalam proses *indexing*.

3.3. *Stemming*

Dalam dokumen teks, suatu kata dapat muncul dalam berbagai bentuk varian, walaupun tetap mewakili makna dasar yang sama. Contoh, kata "order" dapat muncul dalam bentuk lain, seperti "ordering", "ordered" orders", dan sebagainya. Dalam proses *information retrieval*, diharapkan untuk menggabungkan varian-varian dari kata-kata tersebut ke dalam bentuk dasar. Proses untuk menggabungkan atau memecahkan dari setiap varian-varian suatu kata menjadi kata dasar disebut *stemming*. Hasil dari *Stemming* disebut *stem* (akar kata) [9]. *Stem* adalah bagian dari kata yang tersisa setelah dihilangkan imbuhannya (misal, awalan dan akhiran). Contoh, kata *connect* merupakan *stem* dari kata-kata variannya, yaitu *connected*, *connecting*, *connection*, dan *connects*.

Algoritma *Stemming* dikategorikan secara garis besar ke dalam algoritma dibawah ini:

- *Succesor Variety Stemmer*
- N-*gram Stemmer*
- *Affix Removal Stemmer*

Disini akan digunakan algoritma *affix removal stemmer*. Algoritma *affix removal* akan membuang sufiks dan prefiks dari *term* menjadi suatu *stem*. *Affix* (afiks) adalah imbuhan. Dari beberapa algoritma *affix removal*, yang paling umum adalah algoritma *Porter* karena efisiensi dan kecepatannya.

3.3.1. Algoritma *Porter*

Algoritma *Porter Stemmer* adalah suatu proses untuk menghilangkan akhiran morfologikal dan infleksional yang umum pada bahasa Inggris.

Rule-rule pada algoritma *Porter* dibagi menjadi 5 fase. Selanjutnya, *rule* tersebut diaplikasikan secara sekuensial satu sama lain sebagai perintah dalam program [1] Algoritma *Porter* dispesifikasikan dalam bahasa *pseudo programming* seperti dibawah ini:

- Suatu variable konsonan direpresentasikan dengan simbol C yang digunakan untuk menunjukkan semua huruf selain huruf a,e,i,o,u, dan selain huruf y.
- Suatu variabel vokal direpresentasikan dengan symbol V
- Suatu *generic letter* direpresentesikan dengan symbol L
- Simbol 1#1 digunakan untuk *empty string*
- Kombinasi dari C,V, dan L digunakan untuk mendefinisikan *patterns*
- Simbol * digunakan untuk menunjukkan nol atau repetisi lebih dari yang diberikan *pattern*
- Simbol + digunakan untuk menunjukkan satu atau lebih repetisi dari *pattern* yang diberikan
- *Parenthesis* yang sesuai digunakan untuk menempatkan variabel sekuen ke operator * dan +
- *Generic pattern* merupakan kombinasi dari *simbol. parenthesis* yang sesuai, dan operator * dan+
- Substitusi *rule* diperlakukan sebagai perintah yang dipisah dengan tanda baca *semicolon*
- *Rule* subtitusi diaplikasikan ke sufiks pada kata
- Pernyataan kondisional *if* diekspresikan sebagai "*if* (*pattern*) *rule*" dan *rule* dieksekusi hanya jika kondisi *pattern* sesuai dengan kondisi kata
- Baris yang dimulai dengan % diperlakukan sebagai komentar
- *Curled bracket* digunakan membentuk perintah gabungan
- Pernyataan "*select rule with longet suffix*" memilih *rule* tunggal untuk dieksekusi pada semua *rule* dalam perintah gabungan. *Rule* yang dipilih adalah yang paling sesuai dengan sufiks terpanjang

Langkah-langkah prosedural di atas diaplikasikan pada kata-kata dalam teks mulai dari langkah 1 sampai langkah 5

- Langkah 1: berhubungan dengan akhiran bentuk jamak dan lampau

  Misal: *plastered* → *plaster*, *motoring* → *motor*
- Langkah 2: berhubungan dengan kecocokan pola pada beberapa akhiran yang umum

  Misal: *happy* → happi, *relational*→ *relate*
- Langkah 3: berhubungan dengan akhiran kata khusus

  Misal: *hopeful* → *hope*
- Langkah 4: mengecek potongan kata pada akhiran yang lebih panjang untuk menangani kata gabungan

  Misal: *revival* → *reviv*, *allowance* → *allow*

- Langkah 5: mengecek jika potongan kata berakhiran karakter vokal dan memperbaikinya supaya sesuai

  Misal: *probate* → probat, *cease* → ceas

3.4. Pemilihaan Kata Terindeks Untuk Menentukan Grup Kata

Untuk menentukan kata yang akan digunakan sebagai kata terindeks digunakan *abstract view*, pendekatan dengan pengidentifikasian grup kata benda. Strategi termudah untuk pemilihan kata terindeks otomatis adalah dengan menggunakan kata benda pada teks. Dalam hal ini diperlukan pengkelasan kata benda yang muncul di teks menjadi komponen pengindeksan tunggal. Hal itu biasa dilakukan dengan mengadaptasikan *noun group*. *Noun group* adalah seperangkat kata benda yang jarak sintaktisnya pada teks (dihitung pada jumlah kata-kata antara 2 kata benda) tidak melebihi *threshold*/batasan yang ditentukan. Sehingga kata bisa dipecah menjadi 2, yaitu *reverse engineering* dan *software development*.

3.5. *Thesaurus*, Konstruksi Struktur Kategorisasi Kata

Kata *thesaurus* digunakan sebagai perbendaharaan kata-kata atau istilah pada bidang pengetahuan tertentu. Perbendaharaan kata-kata ini terdiri dari:

1. *List* kata-kata penting yang diberikan berdasarkan pengetahuannya
2. Untuk tiap kata pada *list*, ada seperangkat kata yang berhubungan diambil dari hubungan sinonimitas dengan berbagai variasinya.

3.5.1. *Thesaurus* Indeks Kata

Biasanya suatu kata tunggal pada *thesaurus* digunakan untuk menunjukkan suatu konsep yang berdasarkan unit *semantic* untuk menyampaikan ide. Saat suatu konsep tidak dapat diekspresikan dengan kata tunggal, grup kata dapat digunakan sebagai gantinya. Misal, kombinasi kata sifat dan kata benda, seperti *ballistic missiles*, untuk menghindari masalah entri biasanya diubah menjadi *missiles*, *ballistic*.

Penggunaan kata jamak *missiles* lebih disukai dari penggunaan kata tunggal *missile*, karena *thesaurus* menampiikan kelas dari benda-benda dan biasanya berupa bentuk jamak. Selain kata tersebut, biasanya diperlukan penambahan entri *thesaurus* dengan definisi atau penjelasan, karena perlu untuk menspesifikasikan arti kata yang lebih akurat pada konteks dari thesaurus tertentu.

3.5.2. *Thesaurus* Kata Berhubungan

Seperangkat kata yang berhubungan pada kata *thesaurus* dapat diinduksi dari pola kejadian antara dokumen. Hubungan tersebut berhierarki dan diindikasikan dengan kata berhubungan yang meluas (*broader*) direpresentasikan dengan BT atau menyempit (*narrower*) direpresentasikan dengan NT. Hubungan BT dan NT didefinisikan sebagai hirarki klasifikasi dimana BT yang berhubungan dengan suatu kelas akan berhubungan juga dengan NT pada kelasnya. Dapat dikatakan kata-kata tersebut saling berhubungan (*related*) direpresentasikan dengan RT. Bentuk *thesaurus* berdasarkan hubungan inilah yang akan digunakan dalam paper ini.

3.6. Pengindeksan

Pengindeksan diperlukan karena teks tidak dapat dimengerti secara langsung oleh pengklasifikasi. Prosedur pengindeksan harus diterapkan pada dokumen pelatihan, validasi, dan pengujian. Suatu teks di dalam dokumen teks akan direpresentasikan ke dalam vektor bobot *term* [dj] = <w1j, ..., w|T|j>, dengan T adalah himpunan istilah (yang terkadang disebut sebagai *term/feature*, di dalam tulisan ini akan mempergunakan frase "himpunan *term*") yang muncul minimal sekali pada minimal satu dokumen dari koleksi dokumen dalam himpunan dokumen pelatihan (Tr), dan $0 \leq wkj \leq 1$ mewakili seberapa besar suatu *term* tk, tk adalah *term* yang ke-k, berperan dalam memberikan informasi dokumen dj. dj adalah dokumen yang ke-j. Himpunan *term* tersebut dipergunakan untuk mengindeks dokumen teks atau merepresentasikan setiap dokumen teks.

Suatu indeks adalah bahasa yang digunakan untuk merepresentasikan dokumen dan permintaan pengguna. Elemen dari bahasa indeks adalah istilah pengindeksan, yang dapat diperoleh dari teks suatu dokumen yang akan digambarkan, atau dapat juga diperoleh secara independen [15]. Menurut Zobel dan Moffat [16], suatu *term* adalah konsep yang teridentifikasi di dalam suatu dokumen. Untuk dokumen teks, setiap kata di dalam dokumen adalah *term*, setelah kata tersebut mengalami proses pembentukan kata dasar (*stemming*) dan transformasi sejenisnya.

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

a. Beragam pengertian terhadap istilah *term*;

b. Beragam cara untuk menghitung bobot suatu *term*.

Dalam *Oracle Text*, pembobotan menggunakan fungsi standar tfidf, yang diperkenalkan oleh Salton dan Buckley, dengan persamaan:

$$tfidf(t_k, d_j) = (t_k, d_j) \cdot \log \frac{|T_r|}{\tau_r(t_k)} \quad (1)$$

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

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

$$w_{kj} = \frac{tfidf(t_k, d_j)}{\sqrt{\sum (tfidf(t_s, d_j))^2}} \quad (2)$$

4. Metode Klasifikasi

Tujuan dari pengklasifikasian teks atau dokumen adalah untuk mengelompokkan serangkaian dokumen kedalam satu kelompok yang telah didefinisikan sebelumnya. Dengan menggunakan *machine learning*, diharapkan dapat mengenali kelas-kelas dari dokumen secara otomatis. Langkah pertama yang harus diambil dalam pengklasifikasian teks adalah transformasi dokumen, menjadi sebuah representasi yang cocok untuk algoritma pembelajaran dan klasifikasi.

Terdapat beberapa metode pembelajaran yang digunakan dalam *machine learning*, antara lain : *decision tree classifier*, *naïve bayes classifier*, dan *nearest neighbour*. Di sini lebih difokuskan pada penggunaan metode *support decision tree classifier* karena hasil yang lebih akurat.

4.1. *Decision Tree Classifier*

Decision tree merupakan salah satu teknik pembelajaran mesin metode *supervised* yang paling banyak digunakan dan digunakan untuk pengenalan pola, dan termasuk dalam pengenalan pola secara statistik. Dalam pengenalan pola secara statistik, suatu pola direpresentasikan melalui d fitur dan dilihat sebagai suatu titik dalam ruang d-dimensi. *Decision tree* termasuk dalam kelompok "*Decision Boundary Construction*" di mana *decision boundary* dibangun secara langsung berdasarkan data pelatihan yang ada [7].

Decision tree merupakan model untuk mengevaluasi suatu fungsi diskrit. Model ini merepresentasikan perhitungan langkah demi langkah, dengan pada tiap langkah nilai suatu variabel ditentukan, dan berdasarkan nilai tersebut aksi selanjutnya dipilih untuk kemudian dievaluasi.

4.2. Representasi *Decision Tree*

Salah satu cara dalam melakukan klasifikasi suatu pola adalah dengan mengajukan pertanyaan, selanjutnya tergantung pada jawaban untuk pertanyaan saat ini. Pendekatan 20-*questions* ini secara

khusus berguna untuk data non-metrik, karena semua pertanyaan dapat dijawab dengan jawaban "ya tidak" yang tidak memerlukan suatu notasi metrik.

Urutan pertanyaan tersebut ditampilkan dalam suatu *directed decision tree* dimana setiap pertanyaan berkorespondensi dengan suatu *node* dalam *tree* tersebut. *Root node* diletakkan paling atas, tersambung oleh sejumlah *link* atau cabang ke *node* yang lain. Hal ini berlanjut hingga percabangan sampai pada terminal atau *leaf node*, yang tidak memiliki cabang, yang berkorepondensi dengan suatu kategori. Proses klasifikasi suatu pola dimulai pada *root*. Cabang-cabang pada *node* tersebut berkorespondensi dengan sejumlah nilai yang mungkin untuk atribut tersebut. Kemudian dipilih cabang yang sesuai ke *node* selanjutnya, proses tersebut diulang lagi untuk *node-node* selanjutnya, di mana tiap *node* dapat dianggap sebagai *root node* dari suatu *subtree*. Hal ini dilanjutkan terus sehingga dicapai suatu *leaf node*, dimana tidak ada kategori yang berkorespondensi lagi.

Kelebihan model ini dibandingkan model pengklasifikasi yang lain, yaitu interpretabilitas yang muncul dalam dua hal. Pertama, dapat memahami dengan mudah suatu keputusan untuk suatu pola tes tertentu sebagai konjungsi dari tiap keputusan sepanjang *path* dari *root* ke *leaf node* yang bersesuaian. Kedua, dapat diperoleh pemahaman yang jelas mengenai kategori-kategori yang ada. Keuntungan lainnya adalah proses klasifikasi yang cepat.

4.3. Algoritma Dasar *Decision Tree*

Umumnya, algoritma yang dikembangkan untuk membangun suatu *decision tree* merupakan variasi dari algoritma dasar yang menggunakan pendekatan *top-down, greedy search* untuk mencari *decision tree* yang tepat dari semua *decision tree* yang mungkin dibangun [10]. Pendekatan ini dikenal sebagai TDIDT (*Top Down Induction of Decision Tree*).

```
[0] ID3(examples,target_attribute,attributes)
[1]     Create a Root node for the tree
[2]     if not all Examples are positive
[3]         if not all Examples are negative
[4]             if Attributes is not empty
[5]                 A ← the attribute that best classifies Examples
[6]                 The decision attribute for Root ← A
[7]                 for each possible value v_i of A
[8]                     Add a new tree branch below Root, corresponding to
                        The test A = v_i
[9]                     Let Examples_vi be the subset of Examples that have
                        value v_i for A
[10]                    if Examples_vi is empty
[11]                            below this new branch add a leaf node with
                                label equal to the most common value of
                                Target_attribute in Examples
[12]                    else
[13]                            below this new branch add the subtree
                                ID3(Examplesvi,Target_attribute,Attributes – {A})
[14]                    end if
[15]                end for
[16]            else
[17]                label Root with label = most common
                    value of Target_attribute in Examples
[18]            end if
[19]        else
[20]            label Root with label = -
[21]        end if
[22]    else
[23]        label Root with label = +
[24]    end if
[25] return Root
[26] end
```

Gambar 1. Algoritma ID3 untuk mempelajari fungsi bernilai boolean[Mit-1997]

5. Hasil Uji Coba

Uji coba terhadap sistem pengklasifikasi ini dilakukan terhadap data *sample* yang terdiri dari 9 kelas dokumen *web*, dengan setiap kelas terdiri atas 9 buah halaman *web*. Dengan uji coba ini diharapkan akan didapatkan pengukuran kemampuan *decision tree classifier* untuk mengklasifikasikan dokumen *testing* dengan benar dengan pengubahan parameter pada masing-masing proses satu-persatu.

Sedangkan data untuk *stopwords* digunakan *stoplist* dari RainBow (libbow) *toolkit* yang dapat diperoleh dari http://www-2.cs.cmu.edu/~mccallum/bow/, yang telah dikonversi dari file berbahasa C, menjadi suatu format yang dapat dibaca oleh aplikasi penelitian ini. *Stopwords* ini berasal dari SMART, yang dibuat oleh Andrew McCallum, terdiri dari 523 kata.

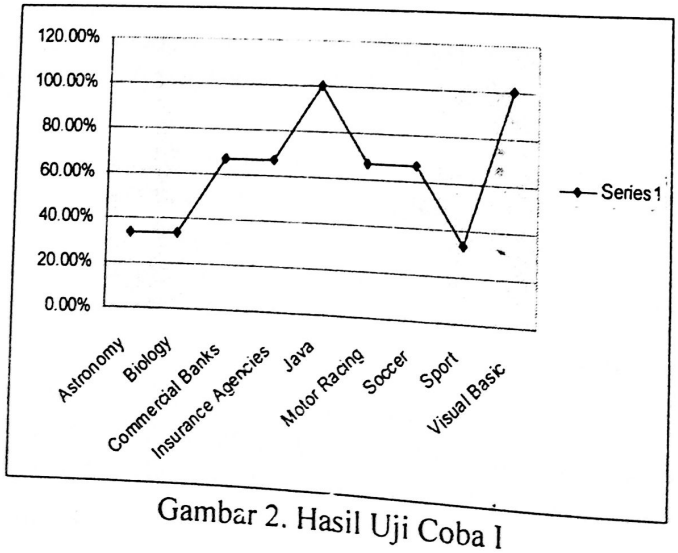
Tabel 1. Deskripsi data set UseNet Collection

Kelas	Total Data
Astronomy	9 file
Biology	9 file
Commescial Banks	9 file
Insurance Agencies	9 file
Java	9 file
Motor Racing	9 file
Soccer	9 file
Sport	9 file
Visual Basic	9 file

5.1. Uji Coba I

Pada uji coba I, dilakukan percobaan terhadap 9 kelas dokumen dengan masing-masing kelas terdiri dari 9 file dengan konfigurasi sebagai berikut:

- Dilakukan penggunaan stemming, penghilangan *stopwords* dan *thesaurus* dengan cara bersamaan.
- Semua data pembelajaran digunakan untuk membuat *rule-rule* pengklasifikasi dan semua data testing digunakan untuk mengetes kemampuan *rule-rule* pengklasifikasi, pertama-tama dengan 3-*fold validation* dan terakhir dengan 10-*fold validation*.
- Proses *training* dilakukan dengan konfigurasi nilai Max *Term* = 20, *Threshold* 40, NT *Threshold* = 0.001, *Term Threshold* = 10.
- Pada uji coba ini dilakukan dengan 3-*fold validation*.

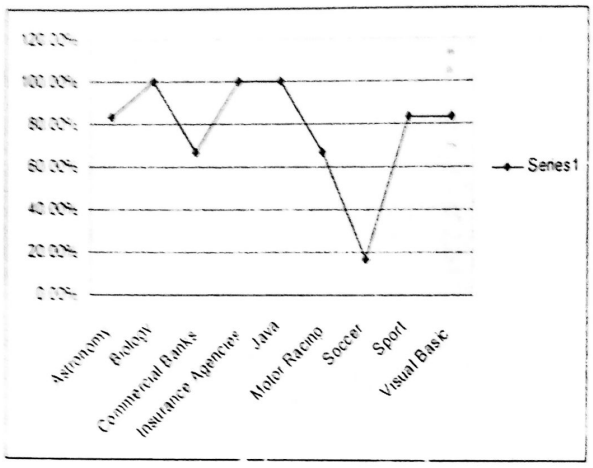


Gambar 2. Hasil Uji Coba I

5.2. Uji Coba II

Pada uji coba II, dilakukan percobaan terhadap 9 kelas dokumen dengan masing-masing kelas terdiri dari 9 file dengan konfigurasi sebagai berikut:

- Hanya dilakukan penggunaan *stemming* pada uji coba ini.
- Semua data pembelajaran digunakan untuk membuat *rule-rule* pengklasifikasi dan semua data *testing* digunakan untuk mengetes kemampuan *rule-rule* pengklasifikasi, pertama-tama dengan 3-*fold validation* dan terakhir dengan 10-*fold validation*.
- Proses *training* dilakukan dengan konfigurasi nilai Max *Term* = 20, *Threshold* 40, NT *Threshold* = 0.001, Term *Threshold* = 10.
- Pada uji coba ini dilakukan dengan 3-*fold validation*.

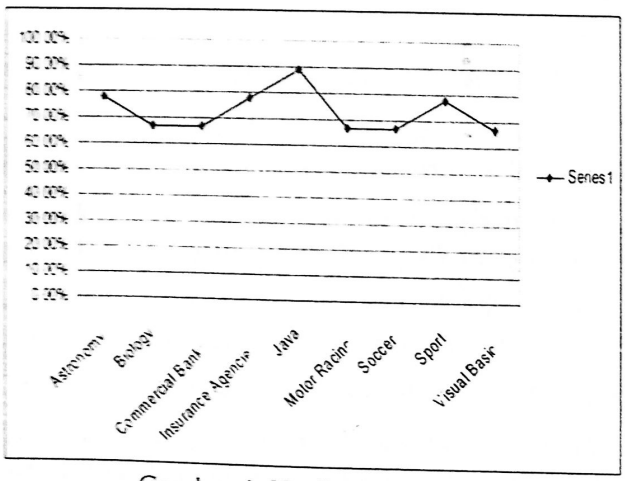


Gambar 3. Hasil Uji Coba II

5.3. Uji Coba III

Pada uji coba III, dilakukan percobaan terhadap 9 kelas dokumen dengan masing-masing kelas terdiri dari 9 file dengan konfigurasi sebagai berikut:

- Hanya dilakukan penggunaan penghilangan *stopwords*.
- Semua data pembelajaran digunakan untuk membuat *rule-rule* pengklasifikasi dan semua data *testing* digunakan untuk mengetes kemampuan *rule-rule* pengklasifikasi, pertama-tama dengan 3-*fold validation* dan terakhir dengan 10-*fold validation*.
- Proses *training* dilakukan dengan konfigurasi nilai Max *Term* = 20, *Threshold* 40, NT *Threshold* = 0.001, *Term Threshold* = 10.
- Pada uji coba ini dilakukan dengan 3-*fold validation*.

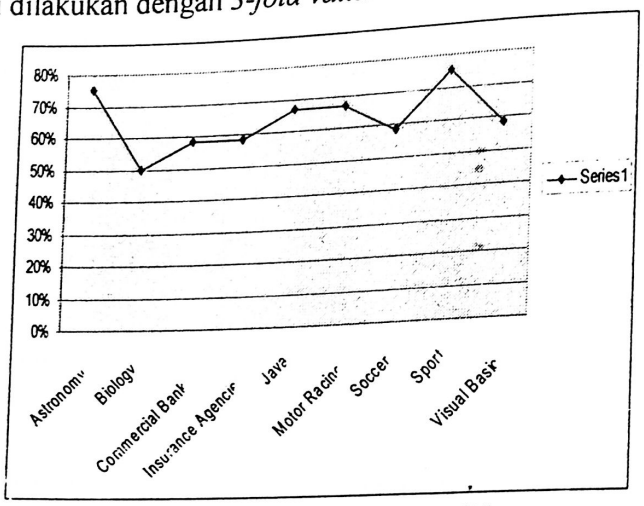


Gambar 4. Hasil Uji Coba III

5.4. Uji Coba VI

Pada uji coba III, dilakukan percobaan terhadap 9 kelas dokumen dengan masing-masing kelas terdiri dari 9 file dengan konfigurasi sebagai berikut:

- Hanya dilakukan penggunaan *thesaurus*.
- Semua data pembelajaran digunakan untuk membuat *rule-rule* pengklasifikasi dan semua data *testing* digunakan untuk mengetes kemampuan *rule-rule* pengklasifikasi, pertama-tama dengan 3-*fold validation* dan terakhir dengan 10-*fold validation*.
- Proses *training* dilakukan dengan konfigurasi nilai Max *Term* = 20, *Threshold* 40, NT *Threshold* = 0.001, *Term Threshold* = 10.
- Pada uji coba ini dilakukan dengan 3-*fold validation*.



Gambar 5. Hasil Uji Coba IV

6. Kesimpulan

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

1. Penggunaan *stemming*, *thesaurus* dan *stopword* pada proses *indexing* mampu meningkatkan kemampuan pengklasifikasi.
2. Peningkatan konfigurasi parameter proses *training* berakibat turunnya kemampuan pengklasifikasi demikian juga sebaliknya. Semakin menurun konfigurasi parameter proses *training*, akan berakibat naiknya kemampuan pengklasifikasi.

Daftar Pustaka

[1] Baeza-Yates, R., Ribeiro-Neto, B, *Modern Information Retrieval*, Addison Wesley, 1999.

[2] Dhillon, S.I., Guan, Y., *Iterative Clustering of High Dimensional Text Data Augmented by Local Search*, 2002.

[3] Duda, R.O., Hart, P.E., Stork, D.G., *Pattern Classification*, John Wiley & Sons Inc, 2001.

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

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

[6] Goodrich, M.T., Mirelli, V., Orletsky, M., Salowe, J., *Decision Tree Construction in Fixed Dimensions: Being Global is Hard but Local Greed is Good*, 1995.

[7] Jain, A. K., Duin, R.P.W. Mao, J., *Statistical Pattern Recognition: A Review*, 1999.

[8] Johnson, D. E., Oles, F.J., Zhang, T., Goetz, T., *A Decision-Tree Based Symbolic Rule Induction for Text Categorization*, IBM System Journal, Vol. 41 No. 3, 2002.

[9] Lam, L. Y., Savio, D., *Learned Text Categorization by Backpropagation Neural Network*, Ph.M. Thesis, Department of Computer Science, Hong Kong University of Science and Technology, 1996.

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

[11] Oracle Corporation, *Oracle Text Reference Guide*, Redwood Shores, CA, 2002.

[12] Oracle Corporation, *Oracle Text Application Developer's Guide*, Redwood Shores, CA, 2002.

[13] Oracle Corporation, *Oracle Text 9.2.0 Technical Overview*, Redwood Shores, CA, 2002.

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

[15] van Rijsbergen, C.J., *Information Retrieval, 2nd edition*, Butterworths, London, 1979.

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