PENGKATEGORIAN ISI BERITA BERBAHASA INDONESIA MENGGUNAKAN ALGORITMA *SYMBOLIC RULE INDUCTION* BERBASIS *DECISION TREE*

Yudhi Purwananto, Diana Purwitasari, Yos Nugroho
Jurusan Teknik Informatika,
Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Kampus ITS, Jl. Raya ITS, Sukolilo – Surabaya 60111, Tel. + 62 31 5939214, Fax. + 62 31 5913804
Email: yudhi@its-sby.edu

ABSTRAK

Pengkategorian teks sangat penting demi manajemen dan temu kembali pengetahuan yang ada pada teks tersebut. Pengkategorian teks yang dilakukan secara manual akan menghabiskan banyak waktu dan biaya. Karena itu diperlukan suatu sistem yang mampu mengkategorikan teks secara otomatis.

Penelitian ini berusaha untuk mengkategorikan teks dengan menggunakan algoritma symbolic rule induction berbasis decision tree. Pengkategorian dilakukan untuk berita berbahasa Indonesia. Dari teks berita tersebut, dipilih fitur-fitur yang relevan untuk masing-masing kategori berdasarkan kriteria Information Gain. Dengan menggunakan fitur-fitur tersebut, dibangun decision tree melalui proses induksi. Untuk meningkatkan akurasi decision tree dilakukan proses pruning. Proses selanjutnya adalah menghasilkan aturan-aturan yang ekivalen secara logis dengan decision tree tersebut dengan memanfaatkan sibling criterion.

Algoritma ini diuji coba dengan menggunakan data berita dari situs Detik. Uji coba dilakukan untuk mengetahui pengaruh dari jumlah fitur, jumlah data, dan nilai maksimum suatu fitur terhadap nilai F_1 dan waktu komputasi. Hasil uji coba menunjukkan bahwa jumlah fitur dan jumlah data pelatihan yang bertambah cenderung akan meningkatkan nilai F_1.

Kata Kunci : *Text Categorization, DTree, Sibling Criterion, Decision Tree, Symbolic Rule Induction*

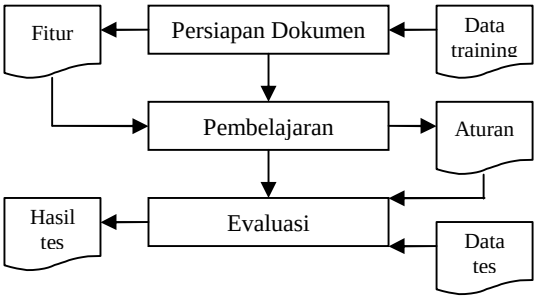
1. PENDAHULUAN

Pengkategorian teks adalah usaha untuk mengkategorikan teks yang belum terkategorikan ke dalam satu atau lebih kategori yang telah didefinisikan sebelumnya berdasarkan informasi yang diperoleh dari data pelatihan yang berupa teks-teks yang telah diketahui kategorinya. Ada beberapa metode yang dapat digunakan untuk melakukan pengkategorian teks secara otomatis. Dari berbagai metode, pendekatan berbasis aturan (*rule based*) memiliki keunggulan dalam hal model yang dihasilkan .

Dalam sistem aturan simbolis, teks direpresentasikan sebagai suatu vektor di mana komponennya adalah frekuensi dari suatu kata dalam teks. Sistem akan menginduksi aturan-aturan dari data pelatihan, dan aturan yang dihasilkan dapat digunakan untuk mengkategorikan data yang lain. Setiap aturan yang dihasilkan pasti terdiri atas suatu kondisi, yang biasanya merupakan konjungsi dari sejumlah kondisi yang lebih sederhana, memiliki implikasi keanggotaan dalam suatu kategori tertentu. Bagian kondisi tersebut membentuk *antecedent* dari aturan dan kesimpulan yang diambil jika kondisi terpenuhi adalah *consequent* dari aturan. Biasanya, *antecedent* dari suatu aturan merupakan kombinasi dari sejumlah tes yng dilakukan terhadap sejumlah komponen dari vektor.

Penelitian ini berusaha membuat sebuah perangkat lunak yang mampu menghasilkan aturan simbolis dengan berbasis pada decision tree. Perangkat lunak ini nantinya akan digunakan untuk mengkategorikan isi berita dalam bahasa Indonesia.

2. ARSITEKTUR



Gambar 1. Arsitektur sistem

Sistem yang akan dibuat memiliki arsitektur seperti pada gambar 1. Terdapat tiga proses utama, yaitu: proses persiapan dokumen, proses pembelajaran, dan proses evaluasi.

3. ENTROPY DAN INFORMATION GAIN

Entropy merupakan salah satu ukuran yang biasa digunakan dalam teori informasi. *Entropy* mengkarakteristikkan ketidakmurnian dari suatu kumpulan data [TOM97]. Jika S adalah suatu kumpulan data, di mana data-data dalam S dapat diklasifikasikan dalam c kategori yang berbeda, maka:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i \qquad (1)$$

Jika semua data dalam S memiliki kategori yang sama maka *entropy* akan sama dengan nol. Namun jika semua kategori memiliki proporsi yang sama dalam S maka *entropy* akan bernilai maksimal.

Information Gain (IG) adalah ukuran yang menyatakan seberapa baik penggunaan suatu atribut untuk mengklasifikasikan data dalam S [TOM97]. IG adalah reduksi nilai entropy yang diperoleh dengan mempartisi data-data dalam S berdasarkan atribut tersebut. Maka IG dari atribut A relatif terhadap S adalah:

$$IG(A) = E(S) - \sum_{v \in Values(A)} \frac{|S_V|}{|S|} E(S_V) \qquad (2)$$

di mana $Values(A)$ adalah himpunan nilai yang mungkin dari atribut A, dan S_V adalah himpunan bagian dari S di mana atribut A memiliki nilai v.

4. PERSIAPAN DOKUMEN

Sebelum suatu dokumen dapat digunakan sebagai data pelatihan, representasinya harus dirubah ke dalam representasi yang sesuai dengan teknik pembelajaran mesin [FAB02].

4.1. PEMROSESAN AWAL

Pada tahap ini dilakukan:

1. Penghilangan kata-kata yang termasuk *stopword*.
2. Proses *stemming*.

Yang dimaksud dengan *stopword* adalah kata-kata yang seringkali memiliki frekuensi yang tinggi, namun tidak dapat dijadikan pembeda antar kategori. Yang termasuk *stopword* antara lain kata hubung, kata sandang, kata ganti, dan lain-lain. Daftar stopword bahasa Indonesia yang dipakai adalah daftar stopword yang sebelumnya telah digunakan oleh Setiono .

Proses *stemming* adalah proses untuk mendapatkan bentuk dasar dari suatu kata yang memiliki imbuhan . Sebagai contoh: kata bekerja, pekerja, mempekerjakan, dikerjakan akan dianggap sebagai kata yang sama, yaitu kerja. Untuk melakukan proses stemming bahasa Indonesia digunakan perangkat lunak yang dikembangkan oleh Nazief .

4.2. PENGINDEKSAN DOKUMEN

Pada tahap ini dilakukan proses pengindeksan terhadap tiap dokumen i dalam data pelatihan sehingga representasinya berubah menjadi $x_i = \{x_{i,1},..., x_{i,D}\}$ di mana D adalah dimensi dari ruang fitur yang digunakan dan $x_{i,d}$ adalah bobot dari fitur d pada dokumen i.

Dengan menggunakan pendekatan *bag of words*, fitur diidentifikasikan sebagai sebuah kata. Pada beberapa penelitian telah ditemukan bahwa representasi yang lebih kompleks seperti frase tidak memberikan hasil yang lebih baik [DAN92]. Metode pembobotan yang digunakan adalah metode tf (term frequency) yang didefinisikan:

$$x_{i,d} = f_{i,d} \qquad (3)$$

di mana $f_{i,d}$ adalah frekuensi kemunculan fitur d pada dokumen i.

4.3. REDUKSI RUANG FITUR

Salah satu cara untuk melakukan reduksi ruang fitur adalah dengan melakukan pemilihan fitur. Pemilihan fitur dilakukan dengan menggunakan metode *Information Gain* (IG) yang memiliki performa baik. Dengan menggunakan metode ini, untuk setiap fitur d dihitung nilai IG yang didapat jika partisi dilakukan dengan melihat ada tidaknya fitur d dalam suatu dokumen. Fitur dengan nilai IG yang tinggi dipilih daripada fitur dengan nilai IG yang rendah. Hal ini diulang untuk setiap kategori c yang ada.

5. PEMBUATAN DECISION TREE

Pembuatan decision tree dibedakan dalam dua tahap. Pada tahap pertama, decision tree ditumbuhkan semaksimal mungkin dari data pelatihan menggunakan algoritma DTree. Pada tahap selanjutnya, dilakukan proses *pruning* dengan menggunakan metode *Error Based Pruning* (EBP) untuk menghindari terjadinya *overfitting*.

5.1. ALGORITMA DTREE

Algoritma DTree [DAV02] adalah algoritma decision tree yang khusus didesain untuk permasalahan pengkategorian teks. Ada dua hal yang membedakan algoritma ini dibanding algoritma decision tree yang lain seperti CART, ID3, dan C4.5, yaitu:

1. Penggunaan *modified entropy* sebagai ukuran ketidakmurnian.
2. Pemanfaatan *sparse structure* dari vektor x untuk mempercepat proses pembuatan decision tree.

Pada algoritma DTree, untuk setiap kategori akan dibangun sebuah decision tree yang menentukan apakah suatu dokumen termasuk dalam kategori tersebut atau tidak. Sehingga untuk setiap kategori, akan ditentukan sebuah label $y_i \in \{0,1\}$ sehingga $y_i = 1$ menunjukkan bahwa dokumen i termasuk dalam kategori tersebut, dan $y_i = 0$ menunjukkan bahwa dokumen i tidak termasuk dalam kategori tersebut.

5.1.1. Induksi Decision Tree

Kebanyakan 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 . Pendekatan ini dikenal sebagai TDIDT (*Top Down Induction of Decision Tree*).

5.1.2. Modified Entropy

Sama seperti algoritma dasar decision tree, algoritma DTree juga melakukan pendekatan *greedy* untuk membangun decision tree. Pada setiap node yang berkorespondensi dengan subset data T dari data pelatihan, dipilih satu fitur f dan satu nilai v sehingga data pada T dipartisi menjadi dua subset $T^1_{f,v}$ dan $T^2_{f,v}$, berdasarkan apakah $x_{i,f} \le v$ sehingga $T^1_{f,v} = \{x_i \in T : x_{i,f} \le v\}$, $T^2_{f,v} = \{x_i \in T : x_{i,v} > v\}$.

Didefinisikan:

$$p^1_{f,v} = P(y_i = 1 \mid x_i \in T^1_{f,v}) \quad (4)$$

$$p^2_{f,v} = P(y_i = 1 \mid x_i \in T^2_{f,v}) \quad (5)$$

$$p_{f,v} = P(x_i \in T^1_{f,v} \mid x_i \in T) \quad (6)$$

Maka transformasi dari estimasi probabilitas $r : [0,1] \to [0,1]$ didefinisikan sebagai berikut:

$$r(p) = \begin{cases} \frac{1}{2}(1 + \sqrt{2p-1} & jika \quad p > 0.5 \\ \frac{1}{2}(1 - \sqrt{1-2p} & jika \quad p \ge 0.5 \end{cases} \quad (7)$$

Modified entropy didefinisikan sebagai berikut:

$$g(p) = -r(p)\log_2(r(p)) - (1-r(p))\log_2(1-r(p)) \quad (8)$$

Untuk setiap kemungkinan *split* (f,v), didefinisikan fungsi *cost* sebagai berikut:

$$Q(f,v) = p_{f,v} g(p^1_{f,v}) + (1 - p_{f,v}) g(p^2_{f,v}) \quad (9)$$

Jika diamati, maka *modified entropy* sebenarnya adalah *entropy* di mana probabilitasnya telah ditransformasikan dengan $r(p)$. Sementara fungsi cost $Q(f,v)$ adalah *modified entropy* setelah dilakukannya partisi.

Modified entropy diperkenalkan untuk menyeimbangkan antara keuntungan penggunaan *classification error* dan *entropy* sebagai ukuran ketidakmurnian .

5.1.3. Sparse Structure

Untuk permasalahan pengkategorian teks, dimensi dari ruang fitur biasanya sangat besar. Di sisi lain, panjang tiap-tiap dokumen, khususnya untuk e-mail dan aplikasi web secara umum cukup pendek. Akibatnya tiap vektor x biasanya memiliki struktur *sparse* (hanya sedikit elemennya yang bernilai lebih dari nol) .

Jika d adalah dimensi dari x, yang menunjukkan jumlah fitur. Dibuat suatu array inclass-count$[1...d][0...V]$ di mana inclass-count[*f*][*v*] adalah jumlah dokumen $x_i \in T$ sedemikian sehingga $y_i = 1$ dan $x_{i,f} = v$. Dibuat juga array total-count$[1...d][0...V]$ di mana total-count[*f*][*v*] adalah jumlah dokumen $x_i \in T$ sedemikian sehingga $x_{i,f} = v$. Waktu yang diperlukan untuk membuat kedua array tersebut adalah $O(|T| \times \bar{l}_T + dV)$ di mana $\bar{l}_T$ adalah rata-rata jumlah elemen yang bukan nol dari $x_i \in T$. Hal ini dapat dilakukan dengan hanya menelusuri komponen-komponen yang bernilai lebih dari nol pada tiap vektor $x_i \in T$.

Setelah langkah ini, dilakukan perulangan melealui $f = 1,...,d$; untuk setiap f, dilakukan perulangan melalui $v = 1,...,V$. Dilakukan penjumlahan untuk mendapatkan jumlah total $x_i \in T$ di mana $x_{i,f} \le v$ dan $y_i = 1$, dan jumlah total $x_i \in T$ di mana $y_i = 1$. Probabilitas $p_{f,v}$, $p^1_{f,v}$, dan $p^2_{f,v}$ dapat diestimasi untuk menghitung $Q(f,v)$. Partisi yang dipilih adalah partisi (f,v) yang memiliki nilai Q terkecil. Langkah ini memiliki kompleksitas $O(dV)$.

Jika diasumsikan bahwa tiap dokumen memiliki panjang yang sama $\bar{l}$, maka total waktu secara kasar yang diperlukan untuk membangun decision tree adalah $O(nh_l\bar{l} + dVM)$, di mana M adalah jumlah node dalam tree, n adalah jumlah dokumen, dan h_l adalah rata-rata depth tree per jumlah dokumen. Faktor yang dominan dalam $O(nh_l\bar{l} + dVM)$ adalah $O(nh_l\bar{l})$. Sebagai perbandingan, algoritma decision tree yang lain akan memiliki kompleksitas paling tidak $O(nh_l d)$, yang biasanya paling tidak sepuluh kali lebih lambat .

5.1.4. Error Based Pruning (EBP)

Dengan metode EBP, data pada suatu node t diasumsikan sebagai suatu sampel statistik dengan jumlah data $N(t)$. Diasumsikan pula bahwa $e(t)$,

jumlah *error* pada node t, mengikuti distribusi Binomial dengan $e(t)$ sukses dari $N(t)$ kali percobaan. Dengan asumsi ini maka dapat diperkirakan suatu *confidence interval* $[L_{CF}(t), U_{CF}(t)]$ untuk *error rate* pada node t. Untuk analisa worst case, maka U_{CF} digunakan sebagai nilai *error rate* yang sebenarnya. Nilai U_{CF} dapat dihitung sebagai berikut [9]:

$$U_{CF}(t) = 1 - beta_{CF}(N(t) - e(t), e(t) + 1) \quad (10)$$

di mana $beta_{\alpha}(a,b)$ adalah nilai yang berasosiasi dengan *percentile* ke α pada distribusi beta dengan parameter a dan b. Setelah nilai U_{CF} didapat, maka nilai tersebut digunakan untuk mendapatkan jumlah error yang sebenarnya, $e'(t)$, pada node t dengan asumsi bahwa *error rate* tersebut digunakan untuk mengklasifikasikan sejumlah data dengan jumlah yang sama dengan jumlah data pada node t.

Proses pruning dilakukan jika $e'(t) \le e'(T_t)$, di mana T_t adalah subtree dengan node t sebagai root node. Proses ini menggunakan pendekatan *bottom-up post-order traversal*.

6. METODE EVALUASI

Pengevaluasian suatu sistem pengklasifikasi biasanya dilakukan secara eksperimental dengan mengukur efektifitasnya, yaitu kemampuannya untuk membuat keputusan klasifikasi yang benar . Keefektifan klasifikasi biasanya diukur dengan dua cara klasik yang juga digunakan dalam sistem IR, yaitu *precision* (π) dan *recall* (ρ) yang diadaptasi untuk permasalahan pengkategorian teks. Untuk menentukan nilai keduanya, digunakan tabel contingency seperti pada tabel 1.

Tabel 1. Tabel contingency

Kategori c_i		Aktual	
		Ya	Tidak
Sistem	Ya	a	b
	Tidak	c	d

Nilai precision dan recall untuk satu kategori tertentu c_i dihitung sebagai berikut:

$$\hat{\pi}_i = \frac{a}{a+b} \quad (11)$$

dan

$$\hat{\rho}_i = \frac{a}{a+c} \quad (12)$$

Sedang untuk mendapatkan nilai *precision* dan recall *keseluruhan* dapat digunakan

- microaveraging

Pada metode *microaveraging*, π dan ρ diperoleh dengan menjumlahkan semua keputusan tunggal dari semua kategori seperti pada tabel 2.2 sehingga:

$$\hat{\pi} = \frac{\sum_{i=1}^{|C|} a}{\sum_{i=1}^{|C|} (a+b)} \quad (13)$$

dan

$$\hat{\rho} = \frac{\sum_{i=1}^{|C|} a}{\sum_{i=1}^{|C|} (a+c)} \quad (14)$$

- macroaveraging

Pada metode *macroaveraging*, π dan ρ diperoleh dengan merata-ratakan nilai π_i dan ρ_i dari tiap kategorinya:

$$\hat{\pi} = \frac{\sum_{i=1}^{|C|} \hat{\pi}_i}{|C|} \quad (15)$$

dan

$$\hat{\rho} = \frac{\sum_{i=1}^{|C|} \hat{\rho}_i}{|C|} \quad (16)$$

Kedua metode di atas akan memberikan hasil yang berbeda. *Microaveraging* akan memberikan bobot yang sama untuk tiap dokumennya, sementara *macroaveraging* memberikan bobot yang sama untuk tiap kategorinya.

Baik *precision* maupun *recall* tidak dapat dipakai secara terpisah satu dengan yang lain. Karena itu diperlukan suatu ukuran yang mengombinasikan nilai π dan ρ . Beberapa ukuran telah dikembangkan, dan salah satunya adalah F_β. Untuk $0 < \beta < +\infty$ maka:

$$F_\beta = \frac{(\beta^2 + 1)\pi\rho}{\beta^2\pi + \rho} \quad (17)$$

di sini β dapat dilihat sebagai bobot relatif dari π dan ρ. Jika $\beta = 0$ maka F_β akan sama dengan ρ. Sementara jika $\beta = +\infty$ maka F_β akan sama dengan π. Biasanya nilai $\beta = 1$ digunakan, nilai ini memberikan bobot yang sama pada π dan ρ [FAB02].

7. HASIL UJI COBA

Data untuk uji coba terdiri atas lima data set, yang semuanya berupa teks berita yang

dikumpulkan dari situs Detik selama tiga bulan mulai tanggal 1 Agustus 2002 sampai 31 Oktober 2002.

Data Detik

- Data Detik merupakan keseluruhan data yang didapat pada situs Detik. Data set ini terbagi dalam enam kategori.

Data DetikSport

- Data set DetikSport merupakan subset dari data set Detik yang termasuk dalam kategori DetikSport. Data set ini terbagi dalam tujuh kategori.

Data DetikFinance

- Data set DetikFinance merupakan subset dari data set Detik yang termasuk dalam kategori DetikFinance. Data set ini terbagi dalam empat kategori.

Data DetikHot

- Data set DetikHot merupakan subset dari data set Detik yang termasuk dalam kategori DetikHot. Data set ini terbagi dalam enam kategori.

Data DetikInet

- Data set DetikInet merupakan subset dari data set Detik yang termasuk dalam kategori DetikInet. Data set ini terbagi dalam tujuh kategori.

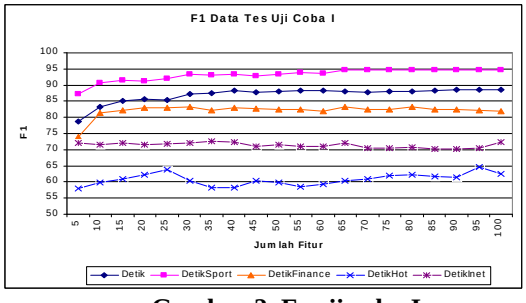
7.1. UJI COBA I

Uji coba I dilakukan untuk menguji pengaruh jumlah fitur yang digunakan dalam tiap kategori terhadap nilai F_1 yang dihasilkan dan waktu komputasi yang dibutuhkan.

Konfigurasi yang digunakan:

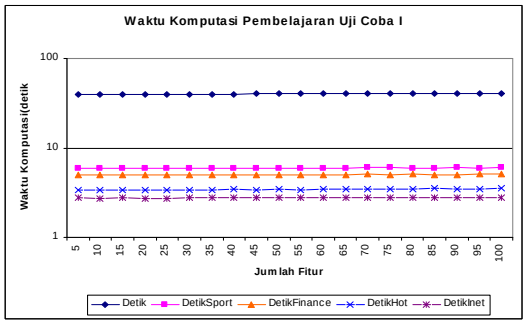
- Dilakukan penghilangan *stopword*.
- Dilakukan proses *stemming*.
- Semua data pembelajaran digunakan.
- Dilakukan proses *pruning*.

Percobaan dilakukan 20 kali dengan jumlah fitur per kategori yang berubah dari 5,10,15,…,100. Pengukuran kinerja didasarkan pada nilai F1 dan waktu komputasi yang dibutuhkan. Nilai F_1 dihitung berdasarkan *microaverage precision* dan *microaverage recall* dari tiap-tiap kategori dalam data tes.



Gambar 2. F_1 uji coba I

Pada data tes seperti pada gambar 2, nilai F_1 menunjukkan kecenderungan peningkatan sejalan dengan penambahan jumlah fitur dengan perkecualian untuk data set DetikHot dan DetikInet. Hal ini dapat terjadi karena distribusi nilai fitur pada data pelatihan dan data tes berbeda. Distribusi yang berbeda menyebabkan partisi yang telah dibentuk berdasarkan data pelatihan menjadi tidak sesuai untuk data tes.



Gambar 3. Waktu komputasi uji coba I

Waktu komputasi pada menunjukkan adanya kecenderungan peningkatan sejalan dengan penambahan jumlah fitur per kategori. Grafik waktu komputasi dapat dilihat pada gambar 3. Peningkatan ini cukup kecil. Peningkatan yang kecil ini karena pemanfaatan *sparse structure* dari vektor fitur.

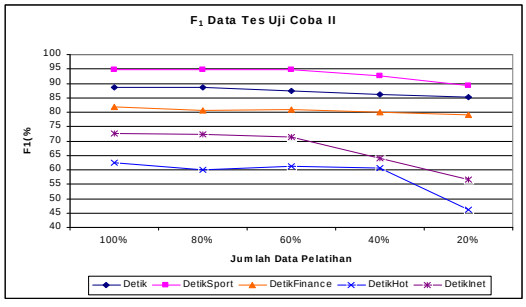
7.2. UJI COBA II

Uji coba II dilakukan untuk menguji pengaruh jumlah data yang digunakan terhadap nilai F_1 yang dihasilkan dan waktu komputasi yang dibutuhkan.

Konfigurasi yang digunakan:

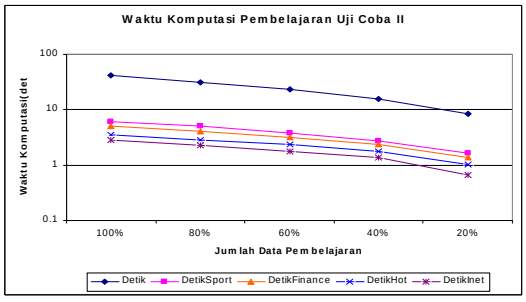
- Dilakukan penghilangan *stopword*.
- Dilakukan proses *stemming*.
- Jumlah fitur per kategori 100.
- Nilai maksimum fitur 3.
- Dilakukan proses *pruning*.

Percobaan dilakukan 5 kali dengan jumlah data yang berubah, yaitu 100%, 80%, 60%, 40%, dan 20% dari total data pelatihan. Data tersebut dihapus secara acak. Pengukuran kinerja didasarkan pada nilai F1 dan waktu komputasi yang dibutuhkan. Nilai F_1 dihitung berdasarkan *microaverage precision* dan *microaverage recall* dari tiap-tiap kategori dalam data tes.



Gambar 4. F1 uji coba II

Pada uji coba II seperti pada gambar 4, terlihat kecenderungan penurunan nilai F_1 pada data tes seiring berkurangnya jumlah data pelatihan. Hal ini terjadi karena berkurangnya jumlah data pelatihan juga berarti berkurangnya kemampuan untuk merepresentasikan keadaan data yang sebenarnya. Karena itu, model yang dihasilkan berdasarkan jumlah data pelatihan yang sedikit juga kurang mampu memodelkan keadaan data yang sebenarnya. Akibatnya nilai F_1 cenderung akan menurun. Namun hal ini tidak selalu terjadi, karena tergantung pada distribusi nilai-nilai fitur yang ada pada data pelatihan itu sendiri.



Gambar 5. Waktu komputasi uji coba II

Waktu komputasi menunjukkan adanya kecenderungan penurunan sejalan dengan berkurangnya jumlah data pelatihan. Penurunan ini disebabkan karena jumlah dokumen yang harus diproses pada tahap persiapan dokumen dan tahap pembelajaran berkurang. Akibatnya waktu komputasi pun ikut berkurang. Grafik waktu komputasi dapat dilihat pada gambar 5.

7.3. UJI COBA III

Uji coba III dilakukan untuk menguji pengaruh nilai maksimum fitur *V* yang digunakan terhadap nilai F_1 yang dihasilkan dan waktu komputasi yang dibutuhkan.

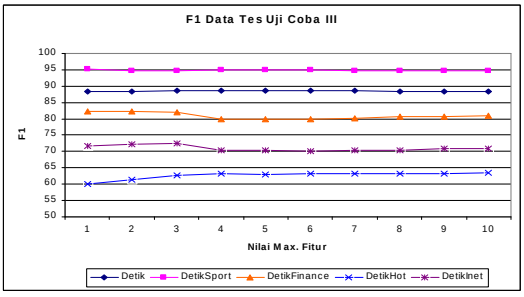
Konfigurasi yang digunakan:

- Dilakukan penghilangan *stopword*.
- Dilakukan proses *stemming*.
- Semua data pembelajaran digunakan.
- Jumlah fitur per kategori 100.
- Dilakukan proses *pruning*.

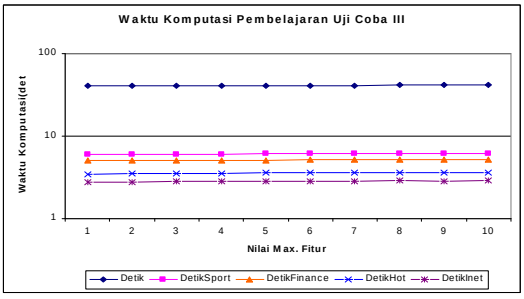
Percobaan dilakukan 10 kali dengan nilai maksimum fitur yang berubah dari 1 sampai 10. Pengukuran kinerja didasarkan pada nilai F1 dan waktu komputasi yang dibutuhkan. Nilai F_1 dihitung berdasarkan *microaverage precision* dan *microaverage recall* dari tiap-tiap kategori dalam data tes.

Pada data tes seperti pada gambar 6, terlihat tidak ada hubungan yang jelas antara penambahan nilai maksimum suatu fitur dengan nilai F_1 yang diperoleh.

Waktu komputasi pada masing-masing data set menunjukkan adanya kecenderungan peningkatan sejalan dengan peningkatan nilai maksimum fitur. Peningkatan waktu komputasi ini cukup kecil. Peningkatan waktu komputasi ini terjadi karena pilihan partisi yang mungkin juga ikut bertambah sehingga pemeriksaan yang harus dilakukan juga ikut bertambah. Grafik waktu komputasi dapat dilihat pada gambar 7.



Gambar 6. F1 uji coba III



Gambar 7. Waktu komputasi uji coba III

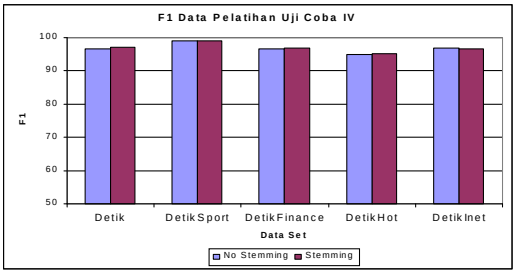
7.4. UJI COBA IV

Uji coba IV dilakukan untuk menguji pengaruh proses stemming terhadap nilai F_1 yang dihasilkan.

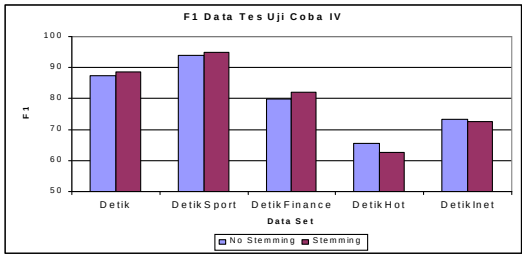
Konfigurasi yang digunakan:

- Dilakukan penghilangan *stopword*.
- Semua data pelatihan digunakan.
- Jumlah fitur per kategori 100.
- Nilai maksimum fitur 3.
- Dilakukan proses *pruning*.

Percobaan dilakukan 2 kali. Percobaan pertama dilakukan tanpa proses stemming sementara percobaan kedua dilakukan proses stemming. Nilai F_1 yang dihasilkan keduanya kemudian dibandingkan.



Gambar 8. F1 data pelatihan uji coba IV



Gambar 9. F1 data tes uji coba IV

8.5 Uji Coba V

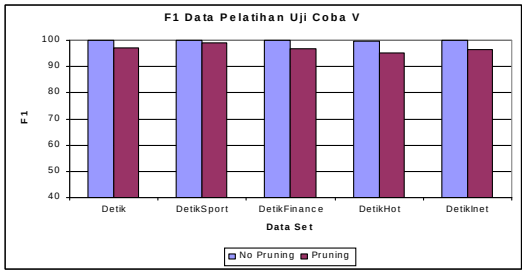
Uji coba V dilakukan untuk menguji pengaruh proses pruning terhadap nilai F_1 yang dihasilkan.

Konfigurasi yang digunakan:

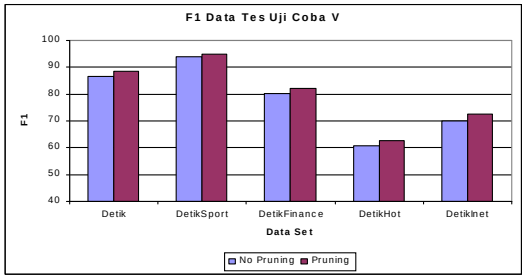
- Dilakukan penghilangan *stopword*.
- Dilakukan proses *stemming*.
- Semua data pelatihan digunakan.
- Jumlah fitur yang digunakan adalah 100.
- Nilai maksimum fitur 3.

Percobaan dilakukan dua kali. Pada percobaan pertama tidak dilakukan proses *pruning*. Sedang pada percobaan kedua dilakukan proses *pruning*. Nilai F_1 yang diperoleh kemudian dibandingkan

Pada data pelatihan seperti pada gambar 10, proses *pruning* menurunkan nilai F_1. Ini terjadi proses *pruning* akan menghilangkan beberapa node pada decision tree, sehingga ketepatan decision tree tersebut dalam memodelkan data pelatihan berkurang. Pada data tes seperti pada gambar 11, proses *pruning* mampu meningkatkan nilai F_1. Ini terjadi karena proses *pruning* mampu meningkatkan kemampuan generalisasi dari decision tree terhadap data yang belum diketahui.



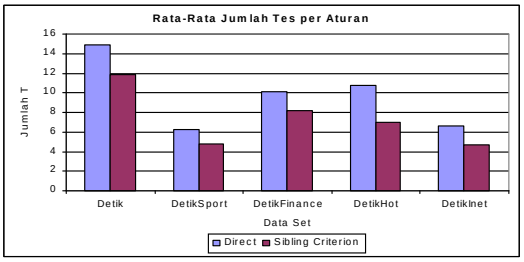
Gambar 10. F1 data pelatihan uji coba V



Gambar 11. F1 data tes uji coba V

7.5. UJI COBA VI

Uji coba VI dilakukan untuk untuk melihat pengaruh penggunaan *sibling criterion* terhadap jumlah tes dalam aturan.



Gambar 12. Jumlah tes per aturan uji coba VI

Konfigurasi yang digunakan:

- Dilakukan penghilangan *stopword*.
- Dilakukan proses *stemming*.
- Semua data pelatihan digunakan.
- Jumlah fitur yang digunakan adalah 100.
- Nilai maksimum fitur 3.
- Dilakukan proses *pruning*.

Percobaan dilakukan dua kali. Pada percobaan pertama, aturan-aturan dihasilkan secara langsung tanpa menggunakan *sibling criterion*. Pada percobaan kedua, aturan-aturan dihasilkan dengan menggunakan *sibling criterion*. Terlihat pada gambar 12 bahwa rata-rata jumlah tes per aturan pada percobaan kedua lebih kecil daripada percobaan pertama. Ini menunjukkan bahwa penggunaan *sibling criterion* mampu menghasilkan aturan yang lebih ringkas.

7.6. UJI COBA VII

Uji coba VII dilakukan untuk memperlihatkan pengaruh pengubahan aturan secara manual terhadap nilai F_1.

Pada uji coba ini, akan ditunjukkan bahwa dengan melakukan perubahan secara manual terhadap aturan-aturan yang dihasilkan, nilai F_1 dapat ditingkatkan. Pengujian dilakukan pada data set DetikSport dengan konfigurasi:

- Dilakukan penghilangan stopword.
- Dilakukan proses *stemming*.
- Semua data pelatihan digunakan.
- Jumlah fitur yang digunakan adalah 5.
- Nilai maksimum fitur adalah 3.
- Dilakukan proses *pruning*.

Dengan konfigurasi seperti di atas, nilai F_1 yang diperoleh adalah 87.18%. Secara khusus, akan dilakukan perubahan terhadap aturan-aturan yang dihasilkan untuk kategori Sepakbola. Aturan-aturan mula-mula yang dihasilkan melalui proses pembelajaran adalah sebagai berikut:

tenis < 1 & balap < 1 & liga > 0 @0.95625001

tenis < 1 & balap < 1 & gol > 0 @0.99683547

tenis < 1 & balap < 1 & klub > 0 @0.84732825
dilakukan penambahan dua aturan sehingga menjadi:
tenis < 1 & balap < 1 & liga > 0 @0.95625001
tenis < 1 & balap < 1 & gol > 0 @0.99683547
tenis < 1 & balap < 1 & klub > 0 @0.84732825
sepak > 0 & bola > 0 @1.00000000
lazio > 0 @ 1.00000000

Sebelumnya, kata *sepak*, *bola*, *dan lazio* harus ditambahkan dulu secara manual pada daftar fitur. Dengan penambahan aturan tersebut, nilai F_1 meningkat menjadi 87.66%.

Terlihat bahwa terjadi peningkatan nilai F_1. Namun perubahan harus dilakukan secara hati-hati, karena perubahan yang dilakukan dapat juga mengakibatkan penurunan nilai nilai F_1. Hal ini dapat ditunjukkan sebagai berikut. Jika aturan-aturan untuk kategori Sepakbola di atas diubah menjadi:

tenis < 1 & balap < 1 & liga > 0 @0.95625001
tenis < 1 & balap < 1 & gol > 0 @0.99683547
tenis < 1 & balap < 1 & klub > 0 @0.84732825
bola > 0 @1.00000000

maka nilai F_1 akan turun menjadi 86.50%.

8. KESIMPULAN DAN SARAN

8.1. KESIMPULAN

3. Pada uji coba I dengan jumlah fitur antara 5 sampai 100, perangkat lunak mampu menghasilkan nilai F_1 rata-rata sebesar 78.89%, nilai F_1 maksimum 94.81%, dan nilai F_1 minimum 57.92% pada data tes. Jumlah fitur yang bertambah cenderung meningkatkan nilai F_1 yang dihasilkan.
4. Penambahan jumlah fitur dari 5 menjadi 100 mengakibatkan peningkatan waktu komputasi rata-rata sebesar 0.25 detik untuk proses pembelajaran. Jumlah fitur yang bertambah meningkatkan waktu komputasi pembelajaran.
5. Pada uji coba II dengan jumlah data pelatihan yang dieliminasi sebesar 20%,40%,60%, dan 80% masing-masing mengakibatkan penurunan nilai F_1 rata-rata sebesar 0.81%, 1.02%, 3.38%, dan 8.78% pada data tes. Jumlah data pelatihan yang berkurang cenderung menurunkan nilai F_1 yang dihasilkan.
6. Jumlah data pelatihan yang dieliminasi sebesar 20%,40%,60%, dan 80% masing-masing mengakibatkan penurunan waktu komputasi rata-rata sebesar 2.52 detik, 4.79 detik, 6.91 detik, dan 9.07 detik pada proses pembelajaran. Jumlah data pelatihan yang berkurang akan menurunkan waktu komputasi pembelajaran.
7. Nilai maksimum fitur yang bertambah cenderung meningkatkan nilai F_1 pada data pelatihan. Namun pada data tes tidak ada hubungan yang jelas antara nilai maksimum fitur dengan nilai F_1.
8. Proses *pruning* akan mengakibatkan peningkatan nilai F_1 rata-rata sebesar 1.85% pada data tes, namun pada data pelatihan akan mengakibatkan penurunan nilai F_1 rata-rata sebesar 2.97%.
9. Penggunaan *sibling criterion* mampu menurunkan jumlah tes pada tiap aturan rata-rata sebesar 2.4 tes per aturan.

8.2. SARAN

1. Pengembangan dapat dilakukan dengan mengintegrasikan kemampuan *incremental learning* untuk decision tree sehingga pengetahuan yang ada pada data baru dapat langsung ikut dimodelkan tanpa melakukan proses pembelajaran dari awal.
2. Untuk meningkatkan nilai F_1 yang diperoleh, dapat digunakan teknik *boosting*. Pada teknik ini, untuk setiap kategori dibuat sejumlah decision tree yang berbeda dan keputusan diambil secara voting.

9. DAFTAR PUSTAKA

1. David E. Johnson, Frank J. Oles, Tong Zhang, Thilo Goetz, *A Decision-Tree Based Symbolic Rule Induction for Text Categorization*, IBM System Journal Vol. 41 No. 3, 2002.
2. Tom M. Mitchell, *Machine Learning*, The McGraw-Hill Companies, Inc, 1997.
3. Fabrizio Sebastiani, *Machine Learning in Automated Text Categorization*, ACM Computing Survey, Vol. 34, No.1, Maret 2002.
4. Ari Novan Setiono, Implementasi Aplikasi Information Retrieval Untuk Pendeteksian dan Klasifikasi Berita Kejadian Berbahasa Indonesia Berbasis Web, Tugas Akhir, Teknik Informatika, Institut Teknologi Sepuluh Nopember Surabaya, 2001.
5. Bobby Nazief, Mirna Adriani, *Confix Stripping Approach to Stemming Algorithm for Bahasa Indonesia*, Fakultas Ilmu Komputer, Universitas Indonesia.
6. Yiming Yang, Jan O. Pedersen, *A Comparative Study on Feature Selection in Text Categorization*, Proceeding, 14th AAAI International Conference on Machine Learning, 1997.
7. Floriana Esposito, Donato Malerba, Giovanni Semeraro, *A Comparative Analysis of Methods for Pruning Decision Trees*, IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 19, No. 5, Mei 1997.
8. Jim Rutledge, Using the Beta Distribution on Confidence Intervals for Proportions.