

IMPLEMENTASI ALGORITMA *WEIGHTED TREE SIMILARITY* DALAM MENENTUKAN PENCARIAN DATA SECARA SEMANTIK

Bilal Abdul Wahid, M Kom

Program Studi Manajemen Informatika

Akademi Manajemen Informatika dan Komputer Bina Sarana Informatika

AMIK BSI Bekasi, Jl. Cut Meutiah No.88 Bekasi

bilal.baw@bsi.ac.id

Intisari— Penggunaan suatu pencarian teks yang lengkap serta pencarian metadata yang sesuai memiliki indikasi kelemahan didalam ketepatan artikel atau data yang dicari. Penelitian ini menjabarkan tentang algoritma kesamaan bobot pohon dikombinasikan dengan metode kesamaan kosinus untuk menghitung kesamaan dalam pencarian data secara semantik. Dalam metadata metode ini dibangun berdasarkan pada pohon berlabel symbol atau node, label dan cabang tertimbang. Bentuk dari Struktur metadata pohon dibangun berdasarkan informasi data semantik seperti taxonomi, ontologi, preferensi, sinonim, homonim dan stemming. Hasil dari pengujian ini adalah ketepatan dalam melakukan pencarian dengan menggunakan algoritma bobot dari pohon berbobot yang lebih baik daripada pencarian teks lengkap dan pencarian dengan metode metadata. Pencarian yang tepat adalah dengan menerapkan Algoritma *Weighted Tree Similarity* guna mencari data secara semantic.

Kata kunci — kemiripan pohon berbobot, pencarian data semantik, kesamaan kosinus.

Abstract— The use of a complete text search and the search for appropriate metadata has indications of weakness in the accuracy of the article or data sought. This study describes the algorithm of tree weight similarity combined with the cosine similarity method to calculate the similarity in semantic data search. In the metadata this method is built based on the tree labeled symbol or node, label and weighted branch. The form of tree metadata structure is built based on semantic data information such as taxonomy, ontology, preference, synonym, homonym and stemming. The result of this test is the accuracy in performing a search using the weight algorithm of a weighted tree that is better than full text search and search by metadata method. The right search is to apply the *Weighted Tree Similarity Algorithm* to search semantic data.

Keywords— similarity of weighted trees, search for semantic data, cosine similarities.

1. Pendahuluan

Melakukan pencarian secara full-text (*full-text search*) merupakan bagian dari tipe pencarian suatu data dokumen yang dilakukan komputer dengan cara menelusuri keseluruhan terhadap suatu isi sebuah dokumen [1]. Cara kerjanya adalah dengan mencari dokumen yang mengandung kata query pengguna. Hasil dari query diurutkan sesuai dengan tingkat kata yang terkandung didalamnya, pada umumnya frekuensi dari kandungan kata diurutkan dari tingkat yang tinggi ke tingkat rendah. Untuk Penelusuran dokumen hanya menggunakan operasi dasar pencocokan kata (*string matching*) tanpa tambahan operasi algoritma lainnya [2]. Dengan metode ini pengguna dapat mengoperasikan dengan mudah, cukup memasukkan kata kunci yang dimaksudkan akan terlihat tampilan antar muka relative lebih sederhana tanpa memasukkan banyak pilihan.

Kekurangan dari metode pencarian full-text [1] ini adalah dari kelemahan linguistik, diantaranya lain tidak dapat mengenali sinonim atau membedakan homonim, serta juga terjadi operasi perbandingan kata yang sangat besar antara kata query pengguna dengan kata di dalam dokumen. Hasil pencarian full-text bisa berupa list yang sangat panjang. Hal Ini dapat terjadi bila di antara banyaknya dokumen mengandung kata yang dicari oleh pengguna. Untuk mencari dokumen yang relevan dengan pengguna, maka harus melihat satu-persatu. Melakukan pencarian dengan metadata biasa (*metadata enabled search*) adalah

tipe pencarian dokumen yang dilakukan komputer dengan menelusuri metadata dokumen [3]. Penggunaan metode ini untuk operasi perbandingan kata jauh lebih sedikit dibandingkan pencarian full-text. Tiap-tiap dokumen diindex dengan dibuat metadatanya. Metadata merupakan 'data dari data' [2], yaitu sekumpulan term terstruktur yang terorganisasi dengan logika AND, OR, sehingga nilai kemiripannya dihitung berdasarkan aritmatika sum of produk atau product of sums sesuai dengan susunan logika AND dan OR yang dipergunakan. Representasi metadata dapat berupa meta tag HTML, file XML atau data dalam

field khusus di database. Metadata ini berfungsi semacam katalog dalam sebuah perpustakaan dan dokumen adalah bukannya. Pada metoda ini query pengguna dapat dilakukan secara lebih spesifik berdasarkan batasan-batasan tertentu. Batasan tersebut seperti pencarian berdasar nama pengarang tertentu, berdasar penerbit tertentu, atau berdasar tahun terbit tertentu Hasil query menjadi lebih sedikit dibandingkan pencarian full-text karena pencarian lebih spesifik, yang diharapkan lebih relevan.

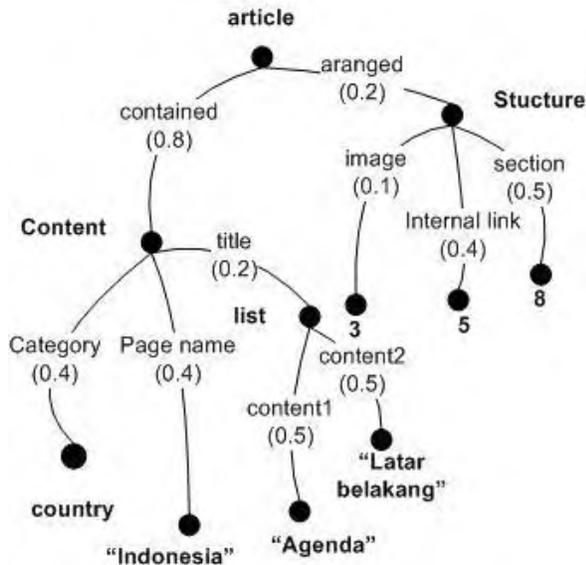
Dalam melakukan pencarian dengan metadata biasa juga memiliki kelemahan linguistik sebagaimana pencarian full-text [1]. Kelemahan lainnya ada pada tambahan operasi pembangkitan metadata yang dapat dilakukan secara manual atau otomatis [3]. Pada dasarnya pencarian dengan metadata adalah memfilter dokumen yang relevan menggunakan logika AND, OR dan pemilihan kata yang tepat. Hal ini menyulitkan karena mencari kata yang tepat terdapat masalah linguistik, antara lain: stemming, prefix, sufix, infix, homonim, dan sinonim.

Dalam pencarian semantik yang menggunakan algoritma *weighted tree similarity*, metadata disusun berdasarkan tree yang memiliki node berlabel, cabang berlabel serta berbobot. Struktur metadata tree disusun berdasarkan informasi semantik semacam taksonomi, ontologi, preference, sinonim, homonim, dan stemming. Sehingga metadata yang digunakan dapat lebih merepresentasikan isi sebuah artikel, serta hasil pencarian dapat lebih tepat (*precision*).

Algoritma *weighted tree similarity* juga memiliki keunikan karena memiliki representasi tree yang berbeda dengan yang lain. Tree yang dipergunakan memiliki node berlabel, cabang berlabel serta berbobot. Untuk merepresentasikan tree dalam algoritma ini secara serial dipergunakan standar *Weighted Object Oriented RuleML (WOORuleML)* yang sesuai dengan standar *Extended Markup Language (XML)*. Representasi ini dapat berfungsi sebagai metadata dokumen yang terindeks. Cabang yang berlabel memberikan pemahaman lebih kepada label nodenya. Begitu pula bobot cabang memungkinkan

memberikan tingkat kecenderungan kepada cabang tertentu lebih dari yang lain.

Pencarian semantik dengan algoritma weighed tree similarity menggunakan algoritma penghitung kemiripan semantik antara dua tree berbobot. Algoritma ini telah diterapkan untuk mencocokkan transaksi ebusiness [4], pencarian obyek pembelajaran [5], virtual market untuk jejaring listrik [6], transaksi kendaraan roda empat [7], estimasi biaya proyek [8], pencarian informasi yang tepat untuk handheld device [9], dan audit otomatis dokumen the International Organization for Standardization (ISO) [10]. Pengembangan algoritma yang diusulkan dalam makalah ini adalah pencocokan khusus untuk subtree khusus term dengan metode cosine similarity. Subtree khusus ini diberi nama subtree keywords. Pengembangan ini mengembangkan makalah sebelumnya [11] yang memperbaiki performa algoritma dalam pencocokan leaf node tree.



Gambar 1: Contoh representasi tree

- 2). Cabang berlabel
- 3). Cabang berbobot
- 4). Label dan bobot ternormalkan.

Didalam label terurut sesuai abjad dari kiri ke kanan, jumlah bobot dalam cabang setingkat subtree yang sama adalah 1. Untuk merepresentasikan tree tersebut digunakan Rule ML versi Object Oriented Modelling, yang disebut Weighted Object Oriented RuleML (WOORuleML) yang mengacu pada standarisasi XML. Contohnya bisa dilihat pada Gambar 2. Pada Gambar 2 terdapat beberapa simbol, adapun keterangan dari simbol-simbol ini antara lain:

- a. <cterm> = keseluruhan tree
- b. <opc> = root dari tree
- c. <ctor> = node label dari root
- d. <_r> = role dari setiap arch/edge dan memiliki beberapa atribut yaitu n mewakili label dan w yang mewakili bobot/weight
- e. <ind> = label untuk role

Subtree dari sebuah role memiliki struktur yang sama atau indentik yang diawali dengan <cterm> dan seterusnya seperti pada Gambar 2.

```
<cterm>
<_opc><ctor>Article</ctor></_opc>
<_r n="Content" w="0,8">
<cterm>
<_opc><ctor>Content</ctor></_opc>
<_r n="category" w="0,4"><ind>Country</ind></_r>
<_r n="page name" w="0,4"><ind>Indonesia</ind></_r>
<_r n="title" w="0,2">
<cterm>
<_opc><ctor>List</ctor></_opc>
<_r n="content1" w="0,5"><ind>Agenda</ind></_r>
<_r n="content2" w="0,5"><ind>Latar Belakang</ind></_r>
</cterm>
</_r>
</cterm>
</_r>
<_r n="Structure" w="0,2">
<cterm>
<_opc><ctor>Structure</ctor></_opc>
<_r n="image" w="0,1"><ind>3</ind></_r>
<_r n="internallink" w="0,4"><ind>5</ind></_r>
<_r n="section" w="0,5"><ind>8</ind></_r>
</cterm>
</_r>
</cterm>
```

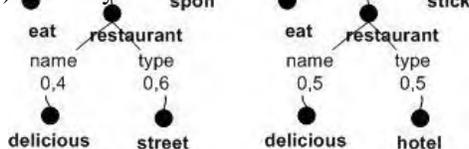
Gambar 2: Representasi tree dalam WOORuleML

2. Metodologi

2.1. Representasi Tree (Weighted Tree Similarity)

Dokumen yang sudah lengkap akan dihitung kemiripannya direpresentasikan dalam sebuah tree yang memiliki karakteristik node berlabel, cabang berlabel dan cabang berbobot. Contoh representasi tree bisa dilihat pada Gambar 1 dimana representasi tersebut menggambarkan representasi query pengguna terhadap sebuah artikel Wikipedia [12]. Di dalam representasi artikel tersebut, pengguna mencari sebuah artikel dengan kategori country, nama halaman Indonesia, subjudul agenda dan latar belakang. Artikel memiliki 3 gambar, 5 link dan 8 subbagian. Keunikan dari weighted tree ini adalah cabang yang berlabel dan berbobot. Pada contoh Gambar 1 pengguna lebih menekankan pencari untuk menemukan ketepatan pencarian berdasarkan cabang content (berbobot 0,8) dibandingkan strukturnya (berbobot 0,2). Penentuan tingkat kepentingan cabang ini terdapat dalam representasi weighted tree. Representasi tree dalam suatu weighted tree mengikuti aturan sebagai berikut [13]:

- 1). Nodonya berlabel



2.2. Penghitungan Kemiripan

Algoritma penghitungan kemiripan antara dua weighted tree ini terdapat di dalam makalah [4], [13]. Gambar 3 menunjukkan contoh dua buah tree T1 dan T2 yang dihitung kemiripannya. Nilai kemiripan tiap pasangan subtree berada diantara interval [0,1]. Nilai 0 bermakna berbeda sama sekali sedangkan 1 bermakna identik. Kedalaman (depth) dan lebar (breadth) tree tidak dibatasi. Algoritma penghitungan dari kemiripan tree tersebut secara rekursif dapat menjelajahi tiap pasang tree dari atas ke bawah mulai dari kiri ke kanan. Algoritma mulai melakukan penghitungan terhadap kemiripan dari bawah ke atas ketika mencapai leaf node selanjutnya nilai dari kemiripan tiap pasang subtree dilevel atas dihitung berdasar kepada kemiripan subtree dilevel bawahnya dilakukan penghitungan, kontribusi bobot cabang juga diperhitungkan. Bobot dirata-rata menggunakan rata-rata aritmatika $(w+w)/2$.

Gambar 3: Contoh kemiripan penghitungan dasar

Dari nilai rata-rata bobot sebuah cabang tersebut dikalikan dengan kemiripan S_i yang diperoleh secara rekursif. Nilai S_i pertama diperoleh berdasar kemiripan leaf node dan dapat diatur nilainya menggunakan fungsi $A(S_i)$. Pada awalnya algoritma weighted tree similarity hanya memberi nilai 1 bila leaf node-nya sama dan 0 bila berbeda [4]. Perumusan penghitungan kemiripan tree ini terdapat di dalam persamaan berikut:

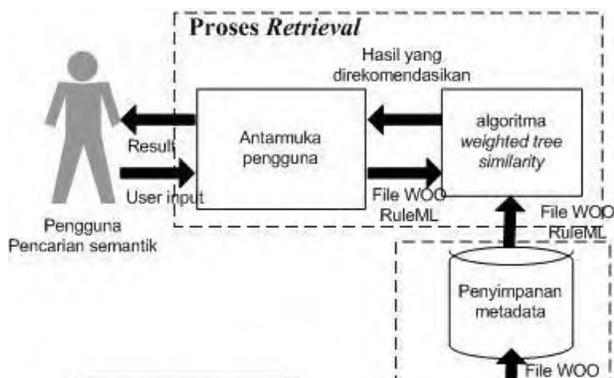
$$\sum (A(S_i)(w_i + w'_i)/2) \quad (1)$$

Nilai dari $A(S_i)$ adalah nilai kemiripan leaf node, w_1 dan w_2 adalah bobot pasangan arc weighted tree. Penilaian $A(S_i)$ analog dengan logika AND sedangkan penilaian bobot pasangan analog dengan logika OR. Di dalam contoh pada Gambar 3 perilaku algoritma dapat dijelaskan sebagai berikut. Pada awalnya dihitung kemiripan node cabang activity yang diperoleh 1. Kemiripan ini dikalikan rata-rata bobot cabang activity $(0,4+0,4)/2$ menghasilkan kemiripan cabang. Algoritma kemudian mencari kemiripan node cabang berikutnya yaitu place. Karena node ini bukan leaf maka algoritma akan turun ke bawah menghitung kemiripan cabang nama Algoritma selanjutnya melakukan penghitungan terhadap kemiripan cabang tipe dan diakumulasi dengan kemiripan cabang name. Akumulasi ini merupakan nilai kemiripan subtree restaurant. Algoritma bergerak ke cabang tool dan akumulasi kemiripan dengan cabang lain yang setingkat menghasilkan kemiripan total.

Merujuk pada makalah [5] pencarian semantik dengan algoritma weighted tree similarity diuraikan dalam Gambar 4. Sistem terdiri atas dua proses, proses indexing dan proses retrieval. Proses indexing terdiri atas pembangkitan metadata dan penyimpanan metadata. Sedangkan proses retrieval berintikan algoritma weighted tree similarity. Pembahasan pembangkitan metadata weighted tree akan diuraikan lebih lanjut di dalam Bagian 3.1, sedangkan pembahasan penggabungan penghitungan weighted tree similarity dengan cosine similarity akan diuraikan di dalam Bagian 3.2.

2.3. Pembangkitan Metadata Tree

Pembangkitan metadata sangat bergantung darimana bentuk representasi tree itu ditetapkan dan ini bersifat kasuistik. Representasi tree yang cocok untuk jual beli motor akan berbeda dengan representasi tree untuk berita online ataupun artikel Wikipedia.



Gambar 4: Pencarian semantik dengan algoritma weighted tree similarity.

Sebagai contoh dalam makalah [12] representasi weighted tree sebuah artikel ditetapkan sebagaimana dalam Gambar 5. Nilai-nilai node dan cabang weighted tree diperoleh dengan mengakses field-field tertentu dalam database Mediawiki. Pencarian semantik dengan algoritma weighted tree similarity dapat diterapkan pada pencarian dokumen, pencarian halaman web, atau pencarian artikel. Jurnal ini mengusulkan pengindeksan koleksi dokumen dengan membangkitkan metadatanya sebelum pengguna melakukan query. Ketika pengguna melakukan query operasi penghitungan kemiripan dilakukan dengan metadata tidak dengan teks dokumen secara langsung. Dalam pembangkitan metadata ini diusulkan perlunya penelusuran teks secara menyeluruh untuk menemukan sekumpulan term representasi konten sebuah dokumen. Dengan kata lain dokumen direpresentasikan sebagai sebuah vector term.

Dalam Gambar 5, representasi tree sebuah artikel wikipedia memiliki sebuah subtree keyword dimana sekumpulan term mewakili konten sebuah dokumen. Sekumpulan term yang mewakili sebuah dokumen ini pada umumnya direpresentasikan sebagai vektor term. Di dalam makalah ini vektor term sebuah dokumen direpresentasikan pula sebagai tree berbobot agar metadata sebuah dokumen menjadi satu kesatuan file WOORuleML. Hal ini dilakukan karena standar metadata WOORuleML digunakan dalam makalah ini untuk representasikan sebuah tree berbobot. Tree berbobot vektor term ini diberi nama subtree keywords.

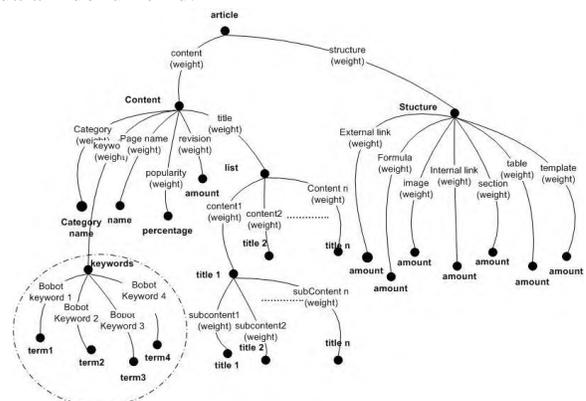
3. Hasil Dan Pembahasan

3.1. Subtree Keywords

Dalam makalah ini konten dokumen direpresentasikan sebagai term vectors [14] dalam bentuk:

$$d = (d_0, d_1, \dots, d_n) \quad (2)$$

Dimana setiap d_k mengidentifikasi term yang terdapat dalam dokumen d .



Gambar 5: Representasi tree

Demikian juga pada query (information request) dari pengguna direpresentasikan dalam term vectors, sehingga dirumuskan:

$$q = (q_0, q_1, \dots, q_n) \quad (3)$$

Dimana setiap q_k mengidentifikasi term yang terdapat pada query q . Sehingga apabila ditentukan bobot (weight) pada setiap term untuk membedakan diantara term yang terdapat dalam dokumen maupun query dapat dituliskan:

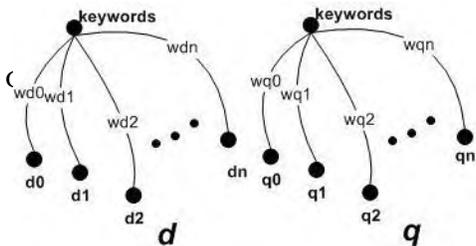
$$w_d = (w_{d0}, w_{d1}, \dots, w_{dn}) \quad (4)$$

dan

$$w_q = (w_{q0}, w_{q1}, \dots, w_{qn}) \quad (5)$$

Dimana w_{dk} merupakan bobot dari term t_k dalam dokumen d , sedangkan w_{qk} merupakan bobot term t_k dalam dokumen q . Dengan demikian subtree keyword query pengguna dan subtree keyword dokumen digambarkan sebagai berikut:

Tampak dalam Gambar 6 weighted tree yang dibangun tidak memiliki label cabang. Untuk membentuk subtree keywords terdapat dua proses utama yang dilakukan, menentukan term nilai leaf node subtree dan menentukan bobot term untuk nilai arc subtree. Untuk menentukan term dilakukan proses tokenizing, stoplist/wordlist, stemming, dan penghitungan term frequency (TF). Dalam proses tokenizing terjadi proses pemotongan dokumen menjadi daftar kata yang berdiri sendiri Di dalam proses stoplist terjadi penyaring kata-kata yang tidak layak untuk dijadikan kata kunci.



Kata-kata yang tidak layak tersebut antara lain kata sambung, kata depan, kata ganti, kata sifat, dan lainnya. Di dalam proses stemming kata dikembalikan ke dalam bentuk dasarnya dengan menghilangkan imbuhan-imbuhan pada kata. Di dalam penghitungan TF dilakukan penghitungan frekuensi kemunculan kata.

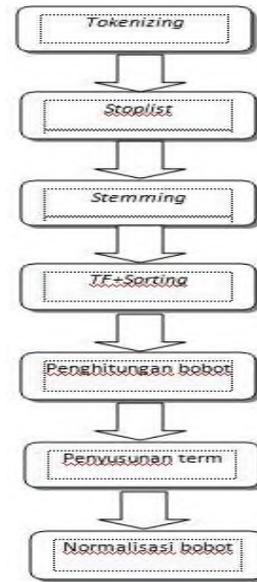
Pembobotan dilakukan dengan memperhatikan TF dalam sebuah dokumen. Penghitungan nilai bobot dilakukan berdasarkan persamaan sebagai berikut:

$$w = TF / T F_{total} \quad (6)$$

Dengan TF_{total} adalah jumlah total TF.

Jumlah term dalam subtree keywords dokumen ditentukan berdasar jumlah bobot term. Bobot term diperoleh mengikuti proses sebagaimana pada Gambar 7. Setelah melalui proses tokenizing, stoplist/word-list, stemming dan TF bobot tiap term ditentukan. Term kemudian diurutkan sesuai bobot kemudian diakumulasikan. Term dipergunakan dan berhenti bila akumulasi bobot term $\geq 0,9$. Nilai bobot total term yang tidak digunakan $\pm 0,1$, cukup kecil dan bisa ditolerir. Dengan demikian jumlah anak cabang sub tree keywords dokumen dapat berubah-ubah tergantung do kumennya.

Demikian pula pada subtree keywords query pengguna. Jumlah term dapat berubah-ubah tergantung term yang diinputkan oleh pengguna.



Gambar 7: Proses pembentukan subtree keywords

Contoh dari proses pembentukan Subtree keywords dokumen sebagai berikut. Setelah melalui proses tokenizing, stoplist, stemming, TF dan sorting, tiap-tiap term dihitung bobotnya dan dihasilkan Tabel 1. Berdasarkan Tabel 1 tampak term yang dipergunakan adalah NMax, Aerox, Mio, dan Vixion karena jumlah akumulasi bobot berada di bawah batas 0,9. Term lainnya tidak dipergunakan untuk merepresentasikan dokumen lebih lanjut.

Bobot ini kemudian dinormalkan agar jumlah bobot dalam subtree ini sama dengan 1. Tabel normalisasi terdapat dalam Tabel 2. Pada contoh ini nilai TF_{total} normalisasi sama dengan 50. Representasi tree pada contoh ini digambarkan dalam Gambar 8.

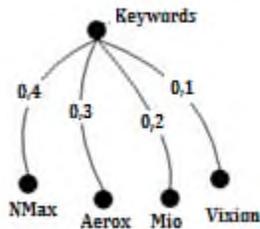
Tabel 1: Contoh penentuan node dan penghitungan bobot subtree keywords dokumen.

Term	TF	Bobot	Akumulasi Bobot
NMax	20	0.350877	0.350877
Aerox	15	0.263158	0.614035
Mio	10	0.175439	0.789474
Vixion	5	0.087719	0.877193
Fino	2	0.035088	0.912281
MXKing	1	0.017544	0.929825
XMax	1	0.017544	0.947368
Byson	1	0.017544	0.964912
Soul	1	0.017544	0.964912
XRide	1	0.017544	0.982456
Total	57		

Tabel 2: Normalisasi bobot

Term	TF	Bobot Normal
NMax	20	0.4
Aerox	15	0.3

Mio	10	0.2
Vixion	5	0.1
Total	50	



Gambar 8: Subtree keywords dokumen contoh

3.2. Penghitungan Kemiripan Gabungan

Algoritma weighted tree similarity pada awalnya bersifat string matching di bagian leaf nodenya. Dengan demikian apabila leaf nodenya sama menghasilkan nilai 1, sedangkan bila berbeda menghasilkan nilai 0. Misalkan terdapat dua buah nilai 70 dan 85, algoritma ini tidak menganggap adanya kemiripan antara dua nilai ini, sehingga similaritynya 0, walaupun secara intuitif dapat dinilai pada kemiripan antara dua nilai tersebut.

Penggunaan pena nganan khusus local similarity untuk leaf node tertentu. Penghitungan kemiripan leaf nodenya dilakukan berdasarkan karakteristik leaf node tersebut. Apakah ia merupakan bilangan, kata, tanggal, atau lainnya. Peningkatan ini mem buat nilai kemiripan leaf node dapat bernilai kontinyu antara 0 sampai dengan 1 yang berarti meningkatkan unjuk kerja algoritma secara umum. Sebagaimana telah diusulkan penelitian [15] dengan menggunakan fuzzy, diusulkan pe nelitian [8] dengan menggunakan rasio, diusulkan makalah [7] untuk dua buah image, dan juga penelitian [10] untuk dua buah dokumen.

Sebagaimana dipaparkan dalam Bagian 3.1, dalam makalah ini konten dokumen direpresentasikan sebagai vektor term. Makalah ini menggunakan penghitungan kemiripan dua vektor yang telah mapan yaitu cosine similarity[2]. Dimana kemiripan dua buah vektor diwakilkan kedalam besar sudut antara dua vektor tersebut.

$$I \text{ Cosine}(q, d) = \frac{q \cdot d}{\|q\| \|d\|} \quad (7) \text{ sebagai berikut:}$$

$$\text{Cosine}(q, d) = \frac{\sum_{k=1}^m w_{qk} \times w_{dk}}{\sqrt{\sum_{k=1}^m (w_{qk})^2} \cdot \sqrt{\sum_{k=1}^t (w_{dk})^2}} \quad (8)$$

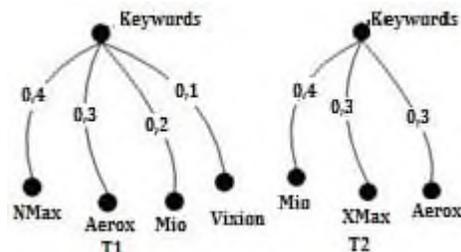
Dengan:

Wij: bobot term j terhadap dokumen i

q : vektor dokumen Q

d : vektor dokumen D

m : dimensi vektor Q dan D



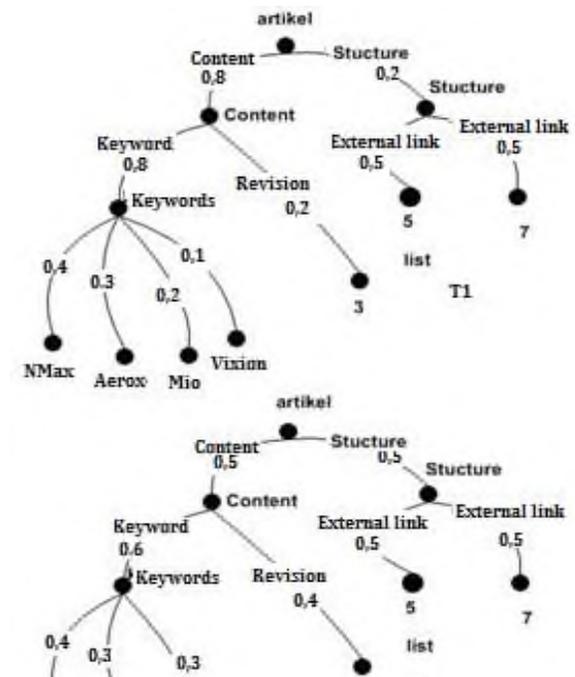
Gambar 9: Subtree contoh T1 dan T2

Dalam Gambar 8 tampak bahwa subtree keywords tidak memiliki label cabang. Hal ini dilakukan untuk memberi fleksibilitas pada tiap term berpindah posisi cabang dalam penghitungan kemiripan. Hal ini mengantisipasi kemungkinan adanya term yang sama pada posisi cabang yang berbeda. Karena kondisi ini berbeda dengan aturan standar weighted tree [5] maka pengembangan yang diajukan pada penelitian ini agar algoritma weighted tree similarity memiliki penanganan khusus untuk subtree keywords. Sedangkan pada penelitian [11] baru diajukan penambahan penanganan khusus penghitungan local similarity leaf node. Subtree ini bersifat khusus baik dari pembentukannya yang tidak memiliki label arc maupun penghitungan kemiripannya yang menggunakan cosine similarity.

Langkah pertama penghitungan adalah penghitungan kemiripan term yang paling tinggi. Operasi perbandingan term ini masih menggunakan operasi string matching, oleh karenanya hanya menghasilkan nilai 1 atau 0 tidak diantaranya. Setelah diketahui pasangan cabang yang memiliki kemiripan 1 kemudian dilakukan perhitungan kemiripan dengan cosine similarity. Komputasi similarity dua weighted tree yang mengandung subtree keywords dipaparkan sebagai berikut. Terdapat penanganan khusus di dalam algoritma untuk subtree keywords. Khusus subtree keywords dipergunakan penghitungan cosine similarity berdasarkan Persamaan 8 sedangkan kemiripan gabungan dihitung dengan perhitungan weighted tree similarity berdasarkan Persamaan 1.

3.3. Subtree Keywords

Contoh komputasi kemiripan weighted tree diterangkan di dalam bagian ini. Pada awalnya diterangkan komputasi kemiripan dua subtree keywords di bagian 4.1. Kemudian baru komputasi kemiripan dua weighted tree yang mengandung subtree keywords dengan weighted tree similarity dipaparkan di bagian 3.4 Terdapat contoh weighted tree query pengguna (T2) dan weighted tree dokumen (T1) sebagaimana Gambar 9. T1 ingin dihitung kemiripannya dengan T2. Komputasi kemiripan cabang menghasilkan Tabel 3 sebagai berikut: Berdasar Tabel 3 dapat diketahui terdapat dua term yang terkandung baik dalam subtree T1 dan T2. Term tersebut adalah Mio dan Aerox.



Gambar 10: Contoh perhitungan total

Dari proses ini dapat dilakukan komputasi kemiripan subtree keyword berdasarkan Persamaan sebagai berikut:

$$\begin{aligned} \text{Cosine}(q, d) &= \frac{(0,4 \cdot 0) + (0,3 \cdot 0,3) + (0,2 \cdot 0,4) + (0,1 \cdot 0)}{\sqrt{0,4^2 + 0,3^2 + 0,2^2 + 0,1^2} \cdot \sqrt{0,4^2 + 0,3^2 + 0,3^2}} \\ &= 0,532 \end{aligned} \quad (9)$$

3. 4. Kemiripan Gabungan

Terdapat contoh weighted tree query pengguna (T2) dan weighted tree dokumen (T1) sebagaimana Gambar 10. T1 ingin dihitung kemiripannya dengan T2. Pada contoh ini subtree keywords sama dengan Gambar 9. Perhitungan subtree ini telah dilakukan pada Bagian 4.1 dan menghasilkan nilai 0,532. Dengan demikian, kemiripan T1 dan T2 di dalam Gambar 9 adalah $\text{Sim}(T1, T2) = 0,78706$.

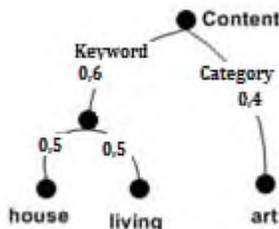
3.5. Uji Coba Dan Evaluasi

Uji coba dilakukan pada Kiwix, versi portabel Wikipedia bahasa Inggris (dapat didownload di <http://www.kiwix.org>). Kiwix berisi 1964 artikel Wikipedia. Pengukuran kinerja pencarian ditentukan dengan suatu perhitungan Precision dengan rumusan sebagai berikut [2]:

$$\text{Precision} = \frac{|R_a|}{|A|} \quad (10)$$

Tabel 3: Penghitungan kemiripan cabang

T1	T2	S
NMax	Mio	0
Aerox	Mio	0
Mio	Mio	1
Vixon	Mio	0
NMax	XMax	0
Aerox	XMax	0
Mio	XMax	0
Vixon	XMax	0
NMax	Aerox	0
Aerox	Aerox	1
Mio	Aerox	0
Vixon	Aerox	0



Gambar 11: Weighted tree pengujian

Dengan Ra adalah jumlah artikel yang relevan dari hasil pencarian, sedangkan A adalah jumlah seluruh artikel hasil pencarian. Nilai precision 1 mengindikasikan kinerja yang tertinggi dan nilai precision 0 terendah.

Pengujian pertama dilakukan pada pencarian full-text dengan relasi OR antara term. Sebagai contoh pada Tabel 4 query pertama bermakna query term house OR living. Pengujian kedua dilakukan pada pencarian menggunakan metadata. Relasi antara term tetap OR dengan tambahan filter AND untuk pemilihan kategori secara spesifik. Sebagai contoh pada Tabel 4 query pertama bermakna query term house OR living AND kategori:arts.

Nilai-nilai hasil pengujian terdapat di dalam Tabel 4. Tampak dari Tabel 4 dengan lima kali pengujian, pencarian dengan metadata menggunakan tambahan filter AND kategori memberikan rata-rata nilai precision yang lebih tinggi dibandingkan pencarian full-text. Hal ini tidak berlaku pada pemilihan kategori yang keliru. Pengujian ketiga dilakukan dengan pencarian menggunakan algoritma weighted tree similarity.

Struktur tree yang dipergunakan terdapat pada Gambar 11 sebagai contoh bentuk tree untuk query pertama (Q1) Tabel 5. Pada query selanjutnya (Q2 s/d Q5) nilai term house dan living digantikan oleh term yang lain sedangkan nilai arts digantikan nilai kategori yang lain mengikuti data dalam Tabel 4. Nilai bobot cabang subtree keywords adalah 1/n dengan n menunjukkan banyaknya query term.

Tree artikel dibangkitkan pula dengan bentuk pada Gambar 11. Pembangkitan subtree keywords mengikuti langkah-langkah pada Bagian 3.1.1. Penghitungan kemiripan gabungan mengikuti langkah-langkah pada Bagian 3.2. Tabel 5 mencatat nilai precision antara tree query pengguna (Q1 s/d Q5) dengan tree artikel pada pencarian menggunakan algoritma weighted tree similarity. Batas suatu artikel dianggap relevan jika precision 0,8. Nilai x/y.

Tabel 4: Hasil pengujian Full-text dan metadata

No	Query terms	Kategori	Precision	
			Full-text	Meta Data
1.	Motorcycle	Transport	0,23	0,4
2.	Animal song	Biology and Medicine	0,08	0,29
3.	Book	Literature	0,25	0,67
4.	Animal kingdom	Biology and Medicine	0,04	0,5
5.	House living	Arts	0,04	0,35
	Rata-rata		0,13	0,44

Tabel 5: Hasil pengujian dengan weighted tree similarity

No	Query	Precision dengan nilai kemiripan _ 0,8
1.	Q1	1/1
2.	Q2	1/1
3.	Q3	1/2
4.	Q4	2/2
5.	Q5	2/2

dalam kolom precision menunjukkan x adalah jumlah artikel yang relevan dari hasil pencarian, sedangkan y adalah jumlah seluruh artikel hasil pencarian. Hasil pada Tabel 5 dan Tabel 4 menunjukkan secara empirik dengan lima kali pengujian, pencarian semantik dengan weighted tree similarity menghasilkan nilai rata-rata precision yang lebih baik (0,9) dibandingkan dengan pencarian full-text (0,13) dan metadata biasa (0,44).

4. Kesimpulan

Hasil dari pencarian data dapat ditunjukkan bahwa penggabungan algoritma weighted tree similarity dengan cosine similarity efektif diimplementasikan untuk pencarian data semantik. Struktur pencarian metadata tree disusun berdasarkan informasi semantik semacam taksonomi, ontologi, preference, sinonim, homonim dan stemming. Hasil uji coba pada artikel Wikipedia menunjukkan bahwa ketepatan (*precision*) pencarian menggunakan algoritma weighted tree similarity lebih tinggi daripada pencarian full-text maupun pencarian dengan metadata biasa.

Penelitian dapat dikembangkan lebih lanjut dengan penggunaan metoda word similarity untuk pencocokan term pada leaf node subtree keywords.

Referensi

- [1] Beall, J.: The Weaknesses of Full-Text Searching. The Journal of Academic Librarianship (September 2008)
- [2] Yates, R.B., Neto, B.R.: Modern Information Retrieval. Addison Wesley Longman Limited (1999)
- [3] Baca, M., et.al: Introduction to Metadata. Getty Research Institute, Los Angeles (2008)
- [4] V.C.Bhavsar, Boley, H., Yang, L.: A Weighted-Tree Similarity Algorithm for Multi-agent System in EBusiness Environments. In: Proceeding Workshop 45 Volume 7, Nomor 1, Januari 2008 : 39–46 on Business Agents and the Semantic Web, National Research Council of Canada, Institute for Information Technology, Fredericton (June 2003) 53–72
- [5] Boley, H., Bhavsar, V.C., Hirtle, D., Singh, A., Sun, Z., Yang, L.: A Match-making System for Learners and Learning Objects. International Journal of Interactive Technology and Smart Education (2005)
- [6] Sarno, R., Yang, L., Bhavsar, V.C., Boley, H.: The AgentMatcher Architecture Applied to Power Grid Transactions. In: Proceeding of the First International

Workshop on Knowledge Grid and Grid Intelligence, Halifax, Canada (2003) 92–99

- [7] Budianto, Sarno, R.: Shape Matching using ThinPlate Splines Incorporated to Extended Weightedtree Similarity Algorithm for Agent Matching in Virtual Market. In: Proceedings International Seminar on Information and Communication Technology. (August 2005)
- [8] Fabianto, E.: Ratio Extended Weighted Tree Similarity Algorithm Applied to Cost Estimate of Software Development. Master's thesis, Program Pasca Sarjana, Fakultas Teknologi Informasi, ITS Surabaya (July 2005)
- [9] Solihin, F., Sarno, R.: Implementasi Arsitektur Agent Matcher Menggunakan Algoritma Extended Weighted Tree Similarity untuk Menyediakan Informasi yang Optimal pada Handheld Device. In: Proceeding Seminar Nasional Pascasarjana VI, Surabaya (August 2006) 38–43
- [10] Sulistyono, W., Sarno, R.: Auto Matching Antar Dokumen dengan Metode Cosine Measure. In: Seminar 46 Nasional Teknologi Informasi dan Komunikasi, Department of Informatics, ITS Surabaya (Mei 2008)
- [11] Yang, L., Ball, M., Bhavsar, V.C., Boley, H.: Weighted Partonomy Taxonomy Trees with Local Similarity Measures for Semantic Buyer-Seller Matchmaking. In: Proceeding of 2005 Workshop on Business Agents and the Semantic Web, Victoria, Canada (May 2005) 23–35
- [12] Rahutomo, F., Sarno, R.: Semantic Search Wikipedia by Applying Agent Matcher Architecture. In: Proceedings International Conference on Information and Communication Technology and System, Department of Informatics, ITS Surabaya (August 2008) 646–653
- [13] Yang, L., Sarker, B.K., Bhavsar, V.C., Boley, H.: A Weighted-Tree Simplicity Algorithm for Similarity Matching of Partial Product Descriptions. In: Proceeding of ISCA 14th International Conference on Intelligent and Adaptive Systems and Software Engineering, Toronto (July 2005) 55–60
- [14] Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison Wesley-Pearson International Edition (2006)
- [15] Setyawan, S.H., Sarno, R.: Fuzzy Logics Incorporated to Extended Weighted-Tree Similarity Algorithm for Agent Matching in Virtual Market. In: Proceeding International Seminar on Information and Communication Technology. (August 2005).