Analisis Perbandingan Metode *Vector Space Model* dan *Weighted Tree Similarity* dengan *Cosine Similarity* pada kasus Pencarian Informasi Pedoman Pengobatan Dasar di Puskesmas

Viko Basmalah Wicaksono
Jurusan Informatika
Universitas Sebelas Maret
Jl. Ir. Sutami 36A Kentingan
Surakarta

viko.jrs@gmail.com

Ristu Saptono
Jurusan Informatika
Universitas Sebelas Maret
Jl. Ir. Sutami 36A Kentingan
Surakarta

ristu.uns@gmail.com

Sari Widya Sihwi
Jurusan Informatika
Universitas Sebelas Maret
Jl. Ir. Sutami 36A Kentingan
Surakarta

sari.widya.sihwi@gmail.com

ABSTRAK

Sistem pencarian merupakan salah satu solusi yang dapat membantu dalam mendapatkan informasi yang diinginkan. Dengan sistem pencarian, proses pencarian informasi akan menjadi lebih efisien. Sistem pencarian informasi pada *ebook* pedoman pengobatan di puskesmas sangat dibutuhkan karena terdapat banyak data penyakit di dalamnya. Dalam mengembangkan sistem pencarian pada pedoman pengobatan di puskesmas, dapat memanfaatkan metode *Vector Space Model* (VSM) atau *Weighted Tree Similarity* (WTS). Penelitian ini membandingkan metode VSM dengan WTS untuk mendapatkan metode terbaik. Selain itu, ditambahkan algoritma *Hamming Distance* untuk mengetahui pengaruh eksekusi waktu sistem.

Penelitian ini menunjukkan bahwa WTS lebih baik dibandingkan VSM, hal ini dapat dilihat pada hasil pengujian, nilai *precision* pada WTS lebih baik dibandingkan VSM. Karena pada metode pencarian yang efektif adalah yang memberikan nilai ketepatan(*precision*) terbaik, meskipun nilai *recall* lebih rendah. Pada pengujian sistem, VSM menunjukkan hasil nilai rata – rata *precision* sebesar 44,82983 % dan *recall* sebesar 99,08165 %. Sedangkan pada WTS nilai rata – rata *precision* sebesar 52,17332% dan *recall* sebesar 98,61761%. Kemudian pada pengujian pakar menunjukkan *precision* WTS dengan rata – rata sebesar 46,675% dan *recall* sebesar 73,6111%. Sedangkan nilai *precision* VSM sebesar 33,6737% dan nilai *recall* sebesar 86,8056%.

Algoritma *Hamming Distance* sangat membantu dalam mempercepat eksekusi sistem. Pengaruh penggunaan algoritma *Hamming Distance* pada VSM memberikan hasil denganrata – rata waktu pengujian adalah 4,512 detik, sedangkan tanpa *Hamming Distance* adalah 9,185 detik. Kemudian Pada metode WTS dengan *Hamming Distance* memberikan hasil rata – rata dengan waktu pengujian adalah 6,042 detik, sedangkan tanpa *Hamming Distance* adalah 14,421 detik.

Kata kunci : *Pedoman Pengobatan Dasar Puskesmas, Sistem pencarian, Vector Space Model, Weighted Tree Similarity*

1. PENDAHULUAN

Dengan banyaknya informasi yang ada di dalam buku Pedoman Pengobatan Dasar di Puskesmas, maka untuk mendapatkan informasi yang dicari dibutuhkan waktu yang relatif lama apabila dilakukan secara manual. Terdapat 109 data penyakit yang masing – masing data memiliki banyak informasi di dalamnya. Apalagi jika terdapat perubahan dan penambahan data, maka informasi yang terbaru akan berbeda dengan informasi yang didapatkan sebelumnya. Dari penjelasan tersebut, permasalahan yang Pada adalah bagaimana cara mendapatkan informasi secara cepat dan akurat. Salah satu solusi dari permasalahan tersebut adalah dengan membuat sistem pencarian.

Sistem pencarian (*Search Engine*) merupakan sistem yang dapat digunakan untuk menemukan informasi yang relevan dengan kebutuhan dari penggunanya secara otomatis dari suatu koleksi informasi [1] . Sistem ini akan menerima masukan berupa kata kunci dari informasi yang akan dicari. Dengan waktu yang relatif lebih singkat akan memberikan hasil beberapa dokumen atau informasi yang relevan terhadap kata kunci yang dimasukkan pengguna.

Dalam sistem temu kembali diperlukan metode yang cocok diterapkan dalam sistem, salah satunya adalah dengan Metode VSM (*Vector Space Model*). VSM adalah suatu metode untuk merepresentasikan sistem temu kembali ke dalam vektor dan memperhitungkan fungsi *similarity* dalam proses pencocokan beberapa vektor. Suatu sistem temu kembali terdiri atas dua bagian, yaitu penyimpanan dokumen dan pemrosesan *query*, baik *query* maupun dokumen-dokumen yang disimpan dinyatakan dalam bentuk vektor [2]. Metode VSM dipilih karena cara kerja model ini efisien, mudah dalam representasi, dan dapat diimplementasikan pada *document-matching* [3].

Selain menggunakan VSM, metode lain untuk pencarian semantik dapat menggunakan WTS (*Weigthed Tree Similarity*). Pencarian semantik dengan *Weighted Tree Similarity* sebelumnya digunakan pada penelitian [5]. *Weighted Tree Similarity* adalah algoritma yang digunakan untuk mengukur tingkat kemiripan dua buah *tree* yang berbobot [4]. Pada *Weighted Tree Similarity*, perhitungan *similarity* pada sub*tree* dihitung dengan metode *Cosine Similarity*. Metode ini telah diterapkan untuk mencocokkan transaksi *e-business*, pencarian obyek pembelajaran, *virtual market* untuk jaringan listrik, transaksi kendaran roda

empat, estimasi biaya proyek, dan pencarian informasi yang tepat untuk *handheld device* [5].

Dalam mempercepat proses pencarian menggunakan metode VSM maupun WTS, digunakan cara menghitung hubungan antar dokumen. Salah satu metode untuk mengetahui hubungan antar dokumen adalah algoritma *Hamming distance*. Pada penelitian [6], penggunaan algoritma *Hamming Distance* diterapkan pada kalimat dalam proses pencarian data dengan bentuk biner agar proses menjadi lebih cepat.

Berdasarkan rincian diatas penulis ingin melakukan analisa perbandingan metode *Vector Space Model* dan *Weighted Tree Similarity* dalam menentukan hasil akurasi sistem pencarian Informasi Pedoman Penyakit Dasar di Puskesmas. Selain itu, dianalisa pengaruh penggunaan algoritma *Hamming Distance* dalam waktu eksekusi pencarian.

2. LANDASAN TEORI

2.1. *Text Preproccessing*

Tahapan *Preproccesing* merupakan tahap persiapan yang dilakukan untuk menyiapkan dokumen sebelum diolah. Penggunaan *text preproccesing* dilakukan karena dokumen teks tidak dapat diproses langsung oleh algoritma pencarian, sehingga diperlukan proses untuk menghasilkan data numerik yang akan digunakan dalam perhitungan. Tahapan *Text Preproccesing* pada penelitian ini meliputi [7]:

a. Penghapusan format dan *markup* dalam dokumen
b. *Tokenizing*
c. *Filtering*
d. *Stemming*

2.2. *Cosine Similarity*

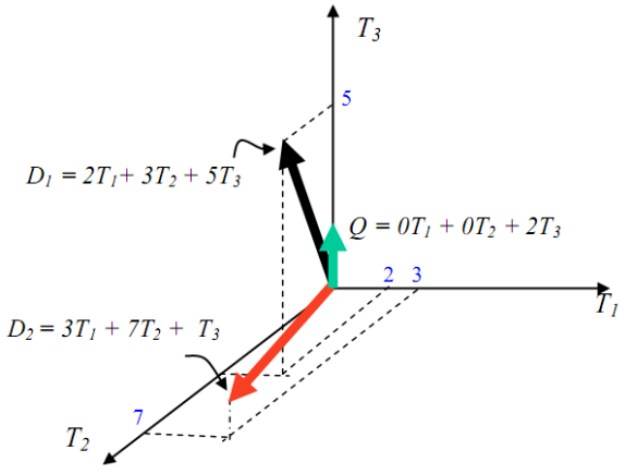
Metode ini digunakan untuk menghitung nilai cosinus sudut antara dua *vector* dan mengukur kemiripan antar dua dokumen [10]. Metode *Cosine Similarity* menggambarkan suatu kesamaan antara vektor *query* dan vektor dokumen dengan dilihat dari sudut yang paling kecil. Perhitungan *Cosine Similarity* dirumuskan pada persamaan (1) berikut ini.

$$sim(d_j, q) = \frac{\vec{d_j} \bullet \vec{q}}{|\vec{d_j}| \bullet |\vec{q}|} = \frac{\Sigma_{i=1}^{t}(w_{ij} \bullet w_{iq})}{\sqrt{\Sigma_{i=1}^{t} w_{ij}^2} \bullet \sqrt{\Sigma_{i=1}^{t} w_{iq}^2}} \quad (1)$$

Dimana:
$\vec{q}$= bobot *query*
$|\vec{q}|$ = panjang *query*
$\vec{d_j}$ = bobot dokumen
$|\vec{d_f}|$= panjang dokumen

2.3. *Vector Space Model* (VSM)

Vector Space Model (VSM) merupakan model yang digunakan untuk mengukur kemiripan antara dokumen dan *query* yang mewakili setiap dokumen dalam sebuah koleksi sebagai sebuah titik dalam ruang [8]. Dalam metode *Vector Space Model* dihitung *weighted* dari setiap *term* yang terdapat dalam semua dokumen dan *query* dari *user*. *Term* adalah kata atau kumpulan kata yang merupakan ekspresi verbal dari suatu pengertian. Penentuan relevansi dokumen dengan *query* dipandang sebagai pengukuran kesamaan antara vektor dokumen dengan vektor *query*. Contoh representasi relevansi antara dokumen dan *query* dapat digambarkan pada Gambar 1. Q merupakan *query* pembanding, D_1 dan D_2 adalah dua dokumen yang akan dibandingkan, sedangkan T_1, T_2 dan T_3 adalah tiga *term* pada dokumen tersebut.



Gambar 1. Representasi dokumen dan *query* pada ruang vektor [8]

Pada perhitungan VSM digunakan pembobotan TF-IDF dan perhitungan nilai *similarity* dengan menggunakan *Cosine Similarity*. Metode TF-IDF adalah cara untuk memberikan bobot hubungan suatu *term* terhadap dokumen. Metode ini menggabungkan dua konsep perhitungan bobot yaitu frekuensi kemunculan kata dalam suatu dokumen dan *inverse* dari frekuensi yang mengandung kata tersebut [9]. Persamaan dalam perhitungan TF-IDF terdapat pada rumusan (2) dan (3) sebagai berikut:

$$W_{(t,d)} = TF_{(t,d)} x\ IDF \quad (2)$$

$$W_{(t,d)} = TF_{(t,d)} x Log \frac{D}{DF_t} \quad (3)$$

Dimana :
W(t,d) : bobot *term* t pada dokumen d
TF (t,d) : total kemunculan *term* t pada dokumen d
D : totalseluruh dokumen
DF_t : total dokumen yang memiliki *term* t

2.3. *Weighted Tree Similarity* (WTS)

Weighted Tree Similarity merupakan salah satu algoritma untuk menentukan tingkat kesamaan dua objek yang disajikan dalam bentuk *tree* [11]. Pada *Weighted Tree Similarity* dokumen yang sudah melalui proses *pre proccessing* direpresentasikan kedalam bentuk *tree*. *Tree* dalam *Weighted Tree* memiliki konsep node atau titik yang berlabel, *arc* atau cabang berlabel, dan *arc* berbobot [5].

Proses pembobotan pada *term* dilakukan dengan memperhatikan TF(*Term Frequency*) dalam sebuah dokumen [5]. Perhitungan pembobotan TF dilakukan dengan persamaan berikut :

$$W = TF/TF_{total} \quad (4)$$

Dimana :
TF = Frekuensi suatu kata
TF_{total} = total frekuensi pada seluruh kata

Dalam perhitungan *similarity* pada *Weighted Tree Similarity*, bobot pada semua bobot cabang berpengaruh pada kemiripan total [4]. Perhitungan dilakukan dengan menghitung rata-rata persamaan aritmatika. Perumusan perhitungan kemiripan *tree* terdapat dalam persamaan (5), yaitu sebagai berikut:

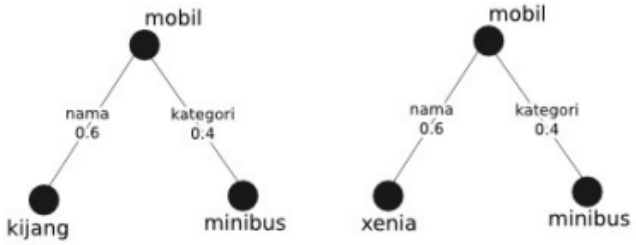
$$\boldsymbol{Sim_{tot}} = \sum(\boldsymbol{S_i} * \boldsymbol{W_i}) \quad (5)$$

Dimana

w_i = bobot cabang *tree* pertama

S_i = kemiripan cabang

Nilai rata-rata bobot hasil perhitungan dikalikan dengan nilai kemiripan cabang (S_i) yang dihasilkan dari perhitungan similarity. Perhitungan *Weighted Tree Similarity* dimulai dengan memberi nilai 1 jika *leaf node*-nya sama dan diberi nilai 0 jika *leaf node*-nya berbeda.



Gambar 2. Contoh perhitungan dasar kemiripan *tree* [4]

Dari Gambar 2 di atas dapat dirumuskan sebagai berikut:

$$\boldsymbol{Sim_{tot}} = (\boldsymbol{Sim}\,[\boldsymbol{kijang}, \boldsymbol{xenia}] * \boldsymbol{0,6}) + (\boldsymbol{Sim}\,[\boldsymbol{minibus}, \boldsymbol{minibus}] * \boldsymbol{0,4})$$

$$\boldsymbol{Sim_{tot}} = (\boldsymbol{0} * \boldsymbol{0.6}) + (\boldsymbol{1} * \boldsymbol{0.4})$$

$$\boldsymbol{Sim_{tot}} = \boldsymbol{0.4}$$

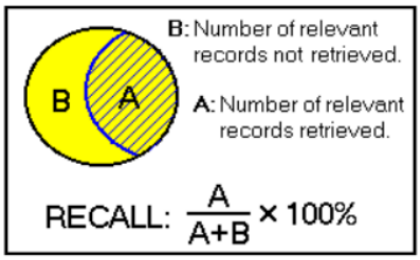
Dari contoh perhitungan pada gambar diatas nilai kemiripan kedua *tree* adalah 0.4. Proses ini akan berulang jika terdapat *subtree* yang lain.

2.4. *Hamming distance*

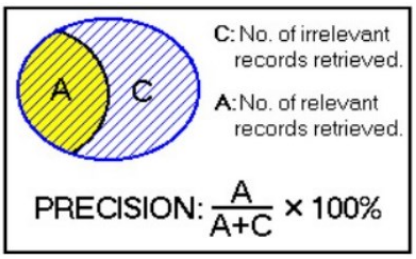
Hamming Distance adalah salah satu algoritma untuk mengukur kedekatan *item*. Jika nilai kedekatannya makin kecil, maka kedua *item* itu semakin dekat dan berlaku sebaliknya. Yang biasanya dibandingkan adalah kata dan bilangan *biner* [5]. Selain digunakan untuk membandingkan kata dan bilangan biner, algoritma *Hamming Distance* juga dapat digunakan pada himpunan kata.

2.5. *Precision* dan *Recall*

Perhitungan *Precision* dan *Recall* merupakan perhitungan yang umum digunakan untuk mengevaluasi hasil pencarian.



Gambar 3. Digram Penjalasan *Recall* [12]



Gambar 4. Digram penjelasan *Precision* [12]

Recall adalah perbandingan antara hasil pencarian yang relevan dengan seluruh data relevan yang ada pada koleksi *database*. Sedangkan *precision* adalah perbandingan antara hasil pencarian yang relevan terhadap semua pencarian yang berhasil di *retrieve*. *Recall* dan *precision* digunakan dalam mengukur tingkat keberhasilan pencarian. Semakin tinggi ukuran *precision* dan *recall*-nya maka semakin bagus strategi pencariannya. Selain itu, suatu sistem dinyatakan efektif apabila hasil penelusuran mampu menunjukkan ketepatan (*precision*) yang tinggi sekalipun perolehan *recall* rendah [12].

3. METODOLOGI PENELITAN

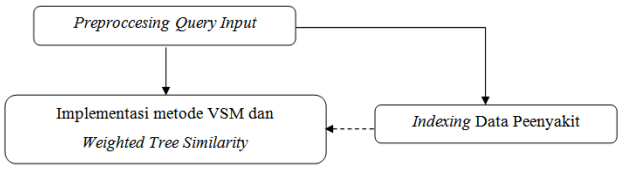
3.1. Studi Literatur

Studi literatur digunakan untuk mendapatkan dasar – dasar referensi yang kuat dalam membantu penyusunan laporan penelitian. Studi literatur dilakukan dengan mengumpulkan literatur yang berkaitan dengan tatalaksana penyakit manusia, metode *Vector Space Model, Weghted Tree Similarity, Stemming* serta *Cosine Similarity*. Sumber – sumber literatur dapat berupa buku, teks, paper, jurnal, karya ilmiah, dan situs – situs yang dapat menunjang penyelesaian penelitian.

3.2. Pengumpulan Data

Tahap pengumpulan data merupakan lanjutan dari studi literatur yang memerlukan data sebagai objek yang akan diolah dan diproses. Data penyakit yang digunakan diambil dari *ebook* Pedoman Pengobatan Dasar di Puskesmas dengan judul "PEDOMAN PENGOBATAN DASAR DI PUSKEMAS" yang diterbitkan oleh Departemen Kesehatan pada tahun 2007.

3.3. Penerapan metode

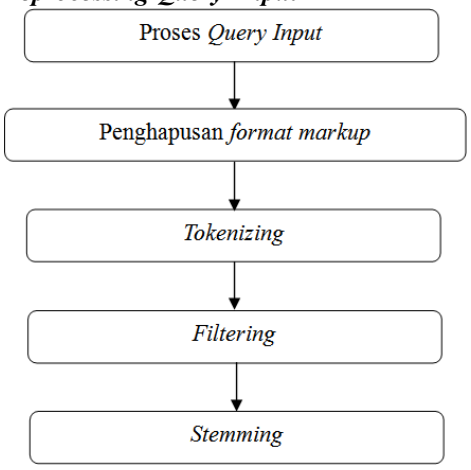


Gambar 5. Proses Penerapan Metode pada Sistem

Pada penelitian ini penerapan metode meliputi Pada penelitian ini penerapan metode meliputi *Preproccesing Query Input* , *Indexing* data penyakit dan Implementasi metode VSM dan *Weighted Tree Similarity*. *Preprocessing query input* merupakan tahapan awal untuk memodelkan *input* sebelum diolah. Pada proses selanjutnya adalah tahapan implementasi metode VSM dan *Weighted Tree Similarity*. Pada tahapan ini akan menghitung proses kemiripan dengan mengambil data penyakit yang sudah diproses dalam *indexing* ke dalam *database*. *Indexing* data

penyakit bertujuan agar pada saat proses penghitungan tidak terlalu lama.

3.3.1. *Preprocessing Query Input*

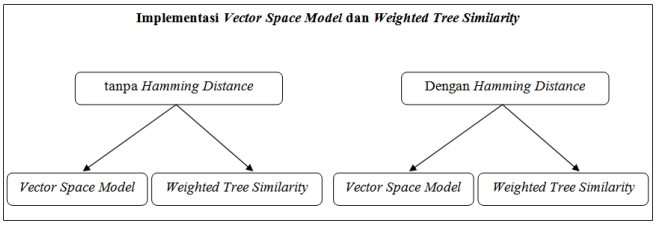


Gambar 6. Proses *Text Preprocessing*

Pada gambar 6 merupakan proses *query input* pada sistem. Proses *query input* dilakukan oleh user, dimana *input* tersebut akan dilakukan tahap *preproccesing*. Pada tahapan ini dilakukan penghapusan *markup* format untuk menghilangkan beberapa format HTML yang tidak digunakan untuk perhitungan. Selanjutnya proses *tokenizing* untuk memotong *input* menjadi *term* kata yang nantinya akan dihitung untuk proses pembobotannya. Kemudian dilakukan proses *Filtering* untuk mengambil beberapa kata penting dan dillanjutkan dengan proses *stopwords removal*. *Stopwords removal* yaitu membuang kata – kata yang dianggap tidak penting dan tidak diperlukan dalam perhitungan. Tahapan terakhir adalah proses *Stemming* untuk memproses kata kembali ke kata dasarnya. Pada penelitian ini digunakan *stemming* Nazief Adriani. Pada tahap *stemming* untuk proses pencarian data, *library* kata dasar yang digunakan adalah *library* dari *term* yang ada pada dokumen. Hal ini dilakukan supaya mempercepat proses *stemming*.

3.3.2. Implementasi metode VSM dan Weighted *Tree Similarity*

Pada proses implementasi metode VSM dan *Weighted Tree Similarity* dapat dilihat pada gambar 7. Proses awal merupakan proses *pre-distance* atau proses penghitungan kedekatan antar dokumen. Langkah ini dibagi menjadi dua jenis, yaitu dengan menggunakan algoritma *Hamming Distance* dan tanpa menggunakan algoritma *Hamming Distance*. Proses ini dilakukan untuk melihat pengaruh penggunaan algoritma *Hamming Distance.* Setelah proses ini selesai dilakukan, kemudian masuk ke dalam tahapan penghitungan kemiripan dengan masing – masing metode.



Gambar 7. *Implementasi* Vector Space Model *dan* Weighted Tree Similarity

3.3.2.1. Implementasi *Hamming Distance*

Pada tahap ini penerapan *Hamming Distance* dilakukan sebelum masuk ke penghitungan pencarian. Implementasi *Hamming Distance* dilakukan dengan melihat hubungan kedekatan antara dokumen dan *query input* dengan melihat *term* apa saja yang terlibat didalamnya. Setelah dilihat kedekatannya, apabila *query* dan dokumen yang akan dibandingkan memiliki persamaan, maka akan dilanjutkan ke dalam proses perhitungan metode selanjutnya. Sedangkan apabila tidak terdapat kedekatan sama sekali antara dokumen dan *query* maka perhitungan akan berhenti dan tidak dihitung ke dalam perhitungan metode selanjutnya.

3.3.2.2. *Vector Space Model*

Pada tahapan ini dijelaskan proses yang terjadi pada metode *Vector Space Model*. Proses tersebut dibagi menjadi tiga, yaitu proses *distance* atau penentuan jarak kedekatan antara dokumen, proses pembobotan dan yang terakhir penghitungan *Similarity*.

Tabel 1. Proses dan Metode pada Vector Space Model

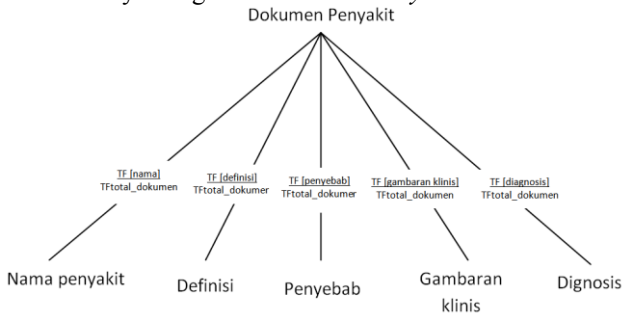
Proses	Vector Space Model
Distance	1
	Hamming distance
Bobot	Pembobotan TF – IDF
Similarity	*Cosine Similarity*

Pada tabel 1, proses VSM dimulai dengan penghitungan jarak (*distance*) untuk mengetahui kedekatan antara *query* dengan dokumen *database*. Pada proses ini dibagi menjadi dua jenis, yaitu menggunakan *Hamming Distance* dan tanpa menggunakan *Hamming Distance* yaitu jarak dianggap 1 atau sama untuk semua dokumen. Langkah selanjutnya adalah proses pembobotan dengan menggunakan TF-IDF. Setelah dilakukan pembobotan, selanjutnya dihitung nilai *similarity* dengan metode *Cosine Similarity*.

3.3.2.3. *Weighted Tree Similarity*

Proses pada WTS (*Weighted Tree Similarity*) langkahnya mirip pada metode VSM, yaitu yang pertama untuk jarak akan dibagi menjadi dua dengan menggunakan *Hamming Distance* dan tanpa menggunakan *Hamming Distance* atau jarak dianggap sama pada *query* dengan semua dokumen. Selanjutnya pada proses pembobotan pada *term* dilakukan dengan pembobotan TF pada tiap parameter.

Pada metode ini, parameter tiap dokumen dibentuk menjadi *tree* data berbobot dan tiap *tree* data dokumen dibuat dengan struktur sama. Nilai bobot tiap parameter diambil dari total *trem frequency* masing – masing parameter dibagi total *term frequency* pada dokumen. Bentuk struktur *tree* pada dokumen penyakit dapat dilihat pada gambar 7. Setelah dihitung bobot pada masing – masing node *tree,* selanjutnya dihitung masing – masing similaritasnya dengan *Cosine Similarity*.



Gambar 8. Susunan *Tree* pada dokumen penyakit

3.3. Pengembangan Aplikasi

Pada penelitian ini dikembangkan sebuah aplikasi pencarian Informasi Pedoman Pengobatan Dasar di Puskesmas dengan menggunakan bahasa pemrograman PHP dan *database* MySql.

3.4. Skenario Pengujian

3.5.1. Pengujian Sistem

Dalam mengevaluasi hasil dari sistem ini dilakukan dengan menghitung *precision* dan *recall* . Penghitungan ini digunakan untuk mengetahui baik atau tidaknya hasil dari pencarian yang digunakan. Pada pemilihan metode terbaik sistem pencarian, dipilih hasil yang memberikan nilai ketepatan (*precision)* lebih tinggi, meskipun nilai *recall* lebih rendah. Selain menghitung *Precision* dan *Recall*, dihitung juga waktu proses yang dilakukan sistem. Perhitungan waktu yang diperlukan digunakan untuk mengetahui efektifas waktu yang digunakan oleh Sistem dari beberapa pemodelan yang digunakan. Prosedur pengujiannya dilakukan dengan mengambil *keyword* dari masing masing *term* di dokumen penyakit yang memiliki frekuensi paling tinggi. Pengujian juga dipisahkan berdasarkan jumlah kata dalam *keyword* yang terbentuk.

3.5.2. Pengujian dengan Pakar

Kemudian yang terakhir dilakukan uji dengan ahli pakar untuk mengetahui apakah sejauh mana relevansi data yang dikembalikan oleh Sistem dengan kata kunci yang dimasukan. Peran pakar disini adalah untuk memilih data mana yang relevan terhadap kata kunci yang dimasukkan. Pakar yang dipilih untuk melakukan pengujian adalah orang yang ahli dibidang penyakit dasar dan kesehatan masyarakat. Kemudian kata kunci yang akan digunakan dalam pengujian didapatkan dari referensi pakar dan penulis. Setelah pakar selesai memilih data mana yang relevan, kemudian akan diproses kembali dalam sistem dengan membandingk VSM dan *Weighted Tree Similarity* untuk mendapatkan nilai *precision* dan *recall.*

4. HASIL DAN PEMBAHASAN

4.1. Deskripsi Data

Data pada penelitian akan menggunakan 109 data penyakit yang ada pada *ebook* Pedoman Pengobatan di Puskesmas yang diterbitkan Kementrian Kesehatan pada tahun 2007Dimana data penyakit tersebut akan disimpan ke dalam *database* yang terdiri dari parameter nama penyakit, definisi, penyebab, gambaran klinis, diagnosis dan penatalaksanaan.

4.2. Penerapan Metode

4.2.1. *Preprocessing Query* Input

Proses *preprocessing query input* bertujuan untuk menyiapkan *query input* yang akan dibandingkan dengan dokumen yang ada di *database*. Pada proses ini dilakukan proses penghapusan format *markup* , *tokeninzing*, *filtering* dan *stemming* . Setelah terbentuk *term* dari *query input* dihitung frekuensi *term* yang akan digunakan sebagai *query* pembanding. Berikut contoh proses *Preproccessing Query Input* dengan *input* user "*Gusi Berdarah dan bengkak"*.

Tabel 2. Penghapusan format *markup* dan *tokenizing*

Query	**Penghapusan Format**	***Tokenizing***
Gusi Berdarah dan bengkak,	gusi berdarah dan bengkak	gusi
		berdarah
		dan
		bengkak,

Proses yang pertama adalah penghapusan format dan *markup* pada *query input* yang dapat dilihat pada tabel 2. Proses ini mengembalikan *query input* dari *user* ke dalam bentuk huruf kecil untuk dilanjutkan ke proses selanjutnya. Selain mengembalikan ke huruf kecil, pada proses ini juga menghapus beberapa format tag yang tidak diperlukan. Selanjutnya, pada proses *tokenizing*, dilakukan pemecahan *query* menjadi beberapa *term* yang dipisahkan berdasarkan spasinya.

Token
gusi
berdarah
dan
bengkak,

→

Filtering
gusi
berdarah
~~dan~~
bengkak~~,~~

→

Hasil *Filtering*
gusi
darah
bengkak

Gambar 9. Proses *Filtering Query Input*

Gambar 9 merupakan proses *filtering.* Pada proses *filtering* bertujuan untuk menghilangkan tanda baca dan symbol yang dianggap tidak penting dan digunakan dalam perhitungan. Pada proses ini juga dilakukan penghapusan *stopwords* yang tidak digunakan dalam perhitungan.

Tabel 3. Hasil *Stemming*

gusi
darah
bengkak

Pada tabel 3 merupakan hasil proses *Stemming* yang bertujuan untuk mengembalikan token yang sudah dipilih kedalam bentuk kata dasarnya. Selanjutnya *term* tersebut yang akan digunakan untuk dihitung frekuensinya.

Tabel 4. Perhitungan Frekuensi *Query Input*

Term	Frekuensi
gusi	1
darah	1
bengkak	1

Langkah yang terakhir dalam proses *preprocessing* adalah menghitung frekuensi dari masing – masing *term query input* . Hasil perhitungan frekuensi dapat dilihat pada tabel 4. Frekuensi kemunculan inilah yang selanjutnya akan digunakan dalam proses pembobotan

4.2.2. Indexing data Penyakit

Tabel 5. Contoh data asli yang belum di*index*

Parameter	Isi dokumen
Nama penyakit	Ginggivitis
Definisi	Ginggivitis adalah Peradangan pada gusi
Penyebab	Karang gusi dan plak
Gambaran Klinis	Gusi bengkak dan mudah berdarah, bau mulut dan nyeri
Diagnosis	Peradangan pada gusi

Sebelum masuk tahap implementasi VSM dan *Weighted Tree Similarity* Dokumen penyakit dokumen penyakit akan dimasukkan dan di-*index* ke dalam *database*. Langkahnya dengan menghitung masing – masing frekuensi *term* dari dokumen kemudian disimpan berdasarkan parameter penyakit. Data pada tabel 5 merupakan contoh data dokumen asli dari penyakit yang di ambil dari buku Pedoman Pengobatan di Puskesmas. Data asli tersebut dimasukkan secara manual oleh admin. Data yang dimasukkan kemudian diolah dan diproses berdasarkan pada proses *text preproccesing* untuk disimpan melalui proses *indexing*.

Contoh Dokumen yang sudah diindex di *Database*

Term	Frekuensi					Total
	NP	D	P	GK	Dg	Frekuensi
ginggivitas	1	1	0	0	0	2
radang	0	1	0	0	1	2
gusi	0	1	1	1	1	4
karang	0	0	1	0	0	1
plak	0	0	1	0	0	1
bengkak	0	0	0	1	0	1
darah	0	0	0	1	0	1
nyeri	0	0	0	1	0	1
bau	0	0	0	1	0	1
mulut	0	0	0	1	0	1

Keterangan :
NP : nama penyakit GK : Gambaran Klinis
D : Definisi Dg : Diagnosis
P : Penyebab

Pada tabel 6merupakan hasil proses *indexing* dokumen penyakit yang ada di *database.* Proses *indexing* menghitung masing – masing frekuensi pada parameter dokumen, karena pada perhitungan *Weighted Tree Similarity* dihitung pada tiap – tiap parameter dan pada VSM perhitungan akan dilakukan dengan menggunakan total frekuensi *term* terhadap dokumen. Selanjutnya data yang sudah di-*index* dihitung ke perhitungan metode selanjutnya.

Pada tahapan *indexing* ditemukan bahwa terdapat *term* yang memiliki banyak frekuensi di semua dokumen. *Term* yang memiliki frekuensi di semua dokumen dapat juga untuk dianggap sebagai *stopword* karena dapat memberikan hasil yang kurang relevan saat perhitungan.

4.2.3. Implementasi Metode VSM dan *Weighted Tree Similarity*

Penerapan implementasi VSM dan *Weighted Tree Similarity* dilakukan untuk mendapatkan metode pencarian yang paling efektif. Dokumen yang akan dibandingkan adalah dokumen penyakit yang sudah diinput dan di*index* ke dalam *database*. Langkah pertama perhitungan dilakukan dengan menghitung masing – masing frekuensi dari dokumen kemudian dilakukan pembobotan terhadap masing – masing data dokumen penyakit. Setelah proses pembobotan selesai kemudian dihitung masing – masing *similarity*nya dengan menggunakan metode *Cosine Similarity*.

Data pada tabel 5 merupakan data dokumen asli dari penyakit yang di ambil dari buku Pedoman Pengobatan di Puskesmas. Data mentah tersebut di*input*kan secara manual oleh admin. Data yang di*input*kan kemudian diolah dan diproses berdasarkan pada proses *text preproccesing* untuk melalui proses *indexing*.

Pada tabel 6 merupakan hasil proses *indexing* dokumen penyakit yang ada di *database.* Proses *indexing* menghitung masing – masing frekuensi pada parameter dokumen, karena pada perhitungan *Weighted Tree Similarity* dihitung pada tiap – tiap parameter dan pada VSM perhitungan akan dilakukan dengan menggunakan total frekuensi *term* terhadap dokumen. Selanjutnya data yang sudah di-*index* dihitung ke perhitungan metode selanjutnya.

4.2.2.1. Penerapan Algoritma *Hamming Distance*

Penggunan *Hamming Distance* dilakukan sebelum masuk pada proses penghitungan metode pencarian. Jadi, proses *Hamming Distance* bertujuan untuk mengetahui bagaimana hubungan kedekatan antara *query input* dengan masing – masing dokumen yang ada pada *database*. Sehingga apabila terdapat dokumen yang sama sekali tidak memiliki kedekatan dengan *query input* tidak akan dilanjutkan dalam perhitungan pencarian. Pada penelitian ini diterapkan penerapan Algoritma *Hamming Distance* pada himpunan kata. Berikut contoh penerapan algoritma *Hamming Distance* pada himpunan kata yang digunakan dalam penelitian ini:

1. Cek panjang himpunan kata. Sebagai contoh berikut :

   Query input = {(gusi),(berdarah)}
   Dokumen *Database* = {(gigi),(berdarah)}

   Dari contoh diatas dilakukan analisa panjang kedua himpunan kata :

   - Pada *Query* input memiliki panjang 2 anggota himpunan, dengan anggota himpunan : gusi dan berdarah.
   - Pada Dokumen *database* memiliki panjang 2 anggota himpunan, dengan anggota himpunan : gigi dan berdarah.

2. Pilih salah satu himpunan sebagai pembanding. Misal kalimat 1 sebagai pembanding.
3. Cek anggota himpunan yang posisinya sama. Jika setiap anggota anggota himpunan pada kedua himpunan memiliki kata yang sama, maka nilainya 0, dan jika berbeda akan diberi nilai 1.

Tabel 13. Contoh penerapan algoritma *Hamming distance*

***Query* Input**	**Dokumen**	**Status**	**Nilai Kedekatan**
gusi		Beda	1
	gigi	Beda	1
berdarah	berdarah	Sama	0
TOTAL			2

4. Tabel 4 merupakan hasil penghitungan, karena kedekatan *Query* dan dokumen adalah 2, maka dapat disimpulkan bahwa *query* dan dokumen cukup mirip.

4.2.2.2. Metode Vector Space Model

Pada implementasi metode VSM akan membandingkan data *query* input yang sudah melalui proses *preproccessing* dengan membandingkan data dokumen penyakit yang ada di *database*.

Tabel 7. Contoh Pembobotan VSM pada *query* dan dokumen penyakit Ginggivitis

Term	**TF**		**IDF**	**Bobo *Term***	
	Q	**P**		**Q**	**P**
ginggivitas	0	2	0,301	0	0,602
radang	0	2	0,301	0	0,602
gusi	1	4	1	1	4
karang	0	1	0,301	0	0,301
plak	0	1	0,301	0	0,301
bengkak	1	1	1	1	1
darah	1	1	1	1	1
nyeri	0	1	0,301	0	0,301
bau	0	1	0,301	0	0,301

Pada Tabel 7 dijelaskan contoh salah satu penghitungan pembobotan TF-IDF dengan *keyword* yang sudah di proses menjadi *query* input. Q merupakan frekuensi dari *query* input sedangkan P merupakan Frekuensi *term* pada dokumen penyakit. Setelah frekuensi Q dan P sudah dihitung, langkah selanjutnya adalah menghitung nilai IDF pada masing – masing dokumen. Dan yang terakhir nilai bobot dihitung dengan mengalikan nila frekuensi masing – masing *term* pada Q dan P dengan nilai IDF.

Setelah proses pembobotan selesai maka didapatkan hasil bobot dari masing – masing *term* terhadap *query* dan dokumen di *database*. Bobot yang sudah didapatkan tersebut digunakan untuk dihitung kemiripannya. Proses penghitungan dilakukan dengan membandingkan kemiripan dari Bobot Q dan Bobot P kedalam bentuk vektor, kemudian dihitung dengan metode *Cosine Similarity*.

Similarity (*Query* , Penyakit)

$$= \frac{\sum(\boldsymbol{Bobot_Q} * \boldsymbol{Bobot_P})}{\sqrt{\sum(\boldsymbol{Bobot_Q}^2)} * \sqrt{\sum(\boldsymbol{Bobot_P}^2)}}$$

$$= \frac{\mathbf{6}}{\mathbf{1{,}73205} * \mathbf{4{,}37927}}$$

$$= \mathbf{0{,}79102}$$

Dari hasil perhitungan *similarity* didapatkan bahwa *similarity* antara *query* yang di*input* kan dengan dokumen penyakit Ginggivitas yang ada di *database* adalah **0,79102**.

4.2.2.3. Metode *Weighted Tree Similarity*

Proses perhitungan pada *Weighted Tree Similarity* dilakukan dengan menghitung masing – masing bobot pada parameter dokumen penyakit. Langkah – langkah perhitungan pada *Weighted Tree Similarity* adalah sebagai berikut:

1. Langkah pertama yaitu membandingkan *term* frekuensi *query* yang sudah di*input* kan, kemudian dibandingkan dengan masing – masing parameter pada dokumen penyakit. Pada Tabel 8,9,10,11 dan 12 dijelaskan pembobotan *term* pada tiap parameter dengan pembobotan TF. Prosesnya hampir sama dengan metode VSM, yaitu Q merupakan frekuensi *term* pada *Query* dan P merupakan frekuensi *term* pada dokumen *database*. Selanjutnya dihitung total frekuensi pada masing – masing Q dan P yang digunakan dalam perhitungan bobot *term* masing – masing *term*.

Tabel 8 Contoh pembobotan TF pada *Query* dan Dokumen Penyakit Ginggivitas parameter nama penyakit

Term	**TF**		**Pembobotan TF**	
	Q	**P**	**Q**	**P**
gusi	1	0	0,33333	0
darah	1	0	0,33333	0
bengkak	1	0	0,33333	0
ginggivitas	0	1	0	1
Total Frekuensi	3	1		

Tabel 9 Contoh pembobotan TF pada *Query* dan Dokumen Penyakit Ginggivitas parameter definisi

Term	**TF**		**Pembobotan TF**	
	Q	**P**	**Q**	**P**
gusi	1	1	0,3333	0,3333
darah	1	0	0,3333	0
bengkak	1	0	0,3333	0
radang	0	1	0	0,3333
ginggivitis	0	1	0	0,3333
Total Frekuensi	3	3		

Tabel 10 Contoh pembobotan TF pada *Query* dan Dokumen Penyakit Ginggivitas parameter penyebab

Term	TF		Pembobotan TF	
	Q	P	Q	P
gusi	1	1	0,3333	0,3333
darah	1	0	0,3333	0
bengkak	1	0	0,3333	0
karang	0	1	0	0,3333
plak	0	1	0	0,3333
Total Frekuensi	3	3		

Tabel 11 Contoh pembobotan TF pada *Query* dan Dokumen Ginggivitas parameter gambaran klinis

Term	TF		Pembobotan TF	
	Q	P	Q	P
gusi	1	1	0,3333	0,1667
darah	1	1	0,3333	0,1667
bengkak	1	1	0,3333	0,1667
nyeri	0	1	0	0,1667
bau	0	1	0	0,1667
mulut	0	1	0	0,1667
Total Frekuensi	3	6		

Tabel 12 Contoh pembobotan TF pada *Query* dan Dokumen Ginggivitas parameter diagnosis

Term	TF		Pembobotan TF	
	Q	P	Q	P
gusi	1	1	0,3333	0,1667
darah	1	1	0,3333	0,1667
bengkak	1	1	0,3333	0,1667
nyeri	0	1	0	0,1667
Total Frekuensi	3	3		

2. Setelah dihitung bobot *term* pada semua parameter, langkah selanjutnya adalah menghitung nilai *similarity* pada tiap parameter terhadap *query* input. Nilai *similarity* dihitung dengan metode *Cosine Similarity* berdasarkan bobot *term* yang sudah dihitung sebelumnya.

Similarity Nama (*Query* , Penyakit)

$$= \frac{\sum(\boldsymbol{Bobot_Q} * \boldsymbol{Bobot_P})}{\sqrt{\sum(\boldsymbol{Bobot_Q}^2)} * \sqrt{\sum(\boldsymbol{Bobot_P}^2)}}$$
$$= \frac{\mathbf{0}}{\mathbf{0,57743 * 1}}$$
$$= \mathbf{0}$$

Similarity Definisi (*Query* , Penyakit)

$$= \frac{\sum(\boldsymbol{Bobot_Q} * \boldsymbol{Bobot_P})}{\sqrt{\sum(\boldsymbol{Bobot_Q}^2)} * \sqrt{\sum(\boldsymbol{Bobot_P}^2)}}$$
$$= \frac{\mathbf{0,01111}}{\mathbf{0,57735 * 0,47141}}$$
$$= \mathbf{0,40825}$$

Similarity Penyebab (*Query* , Penyakit)

$$= \frac{\sum(\boldsymbol{Bobot_Q} * \boldsymbol{Bobot_P})}{\sqrt{\sum(\boldsymbol{Bobot_Q}^2)} * \sqrt{\sum(\boldsymbol{Bobot_P}^2)}}$$
$$= \frac{\mathbf{0,01111}}{\mathbf{0,57735 * 0,47141}}$$
$$= \mathbf{0,40825}$$

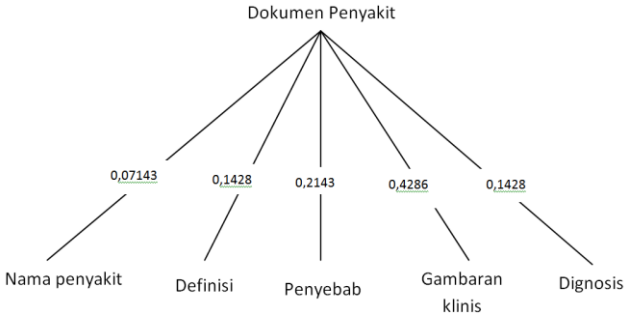
Similarity Gambaran Klinis (*Query* , Penyakit)

$$= \frac{\sum(\boldsymbol{Bobot_Q} * \boldsymbol{Bobot_P})}{\sqrt{\sum(\boldsymbol{Bobot_Q}^2)} * \sqrt{\sum(\boldsymbol{Bobot_P}^2)}}$$
$$= \frac{\mathbf{0,16667}}{\mathbf{0,57735 * 0,40285}}$$
$$= \mathbf{0,70711}$$

Similarity Diagnosis (*Query* , Penyakit)

$$= \frac{\sum(\boldsymbol{Bobot_Q} * \boldsymbol{Bobot_P})}{\sqrt{\sum(\boldsymbol{Bobot_Q}^2)} * \sqrt{\sum(\boldsymbol{Bobot_P}^2)}}$$
$$= \frac{\mathbf{0,01111}}{\mathbf{0,57735 * 0,47141}}$$
$$= \mathbf{0,40825}$$

3. Langkah terakhir pada metode *Weighted Tree Similarity* adalah menghitung nilai *similarity* total dari *query* dan dokumen. Nilai *similarity* dihitung dengan mengalikan nilai *similarity* tiap cabang/ parameter dengan bobot parameter. Bobot parameter didapatkan dari nilai total *term* frekuensi tiap parameter dibagi total frekuensi *term* keseluruhan dokumen. Sehingga didapatkan bobot *tree* parameter seperti pada gambar 10.



Gambar 10. Proses *Filtering Query Input*

4. Setelah didapatkan nilai bobot pada tiap node parameter, selanjutnya dihitung nilai *similarity* total. Langkahnya adalah dengan menghitung nilai total *similarity* pada masing – masing parameter kemudian dikalikan dengan nilai bobot tiap node parameter. Sehingga penghitungan nilai *similarity* total dengan metode *Weighted Tree Similarity* didapatkan sebagai berikut

Similarity Total
= (0*0,7143) + (0,40825*0,1428) + (0,40825*0,2143) + (0,70711*0,4286) + (0,40825*0,1428)
= **0,50715**

Sehingga didapatkan bahwa *similarity* total dengan metode *Weighted Tree Similarity* antara *query* yang di*input* kan dengan dokumen penyakit Ginggivitas yang ada di *database* adalah **0,50715**. Pada hasil penghitungan *similarity* yang diberikan oleh *Weighted Tree Similarity* memberikan nilai *similarity* yang nilainya lebih sedikit dibandingkan VSM. Hal ini kemungkinan dipengaruh oleh nilai bobot paramater *node tree* pada dokumen penyakit yang diambil berdasarkan jumlah *term* yang ada didalamnya. Hal ini bisa dioptimalkan dengan memberikan nilai bobot node yang lebih relevan tiap parameternya, agar nilai *simalirity* lebih tinggi dan hasil lebih relevan.

4.3. Hasil dan Pembahasan

Pada hasil dan evaluasi yang dilakukan, dibagi menjadi 2 kali pengujian, yaitu pengujian dari Sistem dan pengujian dengan pakar.

4.3.1. Pengujian Sistem

4.3.1.1. Penentuan *Keyword* Pengujian

Pada proses pengujian sistem digunakan dengan mengambil *keyword* secara otomatis dari *database*. *Keyword* tersebut diambil dari *term* yang memiliki frekuensi paling tinggi dari masing – masing parameter pada dokumen penyakit. Pengambilan *term* tertinggi diambil kemudian dikombinasikan menjadi sebuah *keyword* yang terdiri dari satu sampai lima kata.

Tujuan dilakukan percobaan ini adalah untuk mendapatkan nilai batasan yang akan digunakan sebagai *threshold* dan juga untuk pengujian sistem. Pada tabel 14 didapatkan hasil dari pembentukan jumlah *keyword* dan banyak kombinasinya. *Keyword* yang terdiri dari 1 kata merupakan *term* dengan frekuensi yang tertinggi dari masing – masing dokumen penyakit. Sedangkan untuk *keyword* yang terdiri dari dua sampai lima kata, merupakan rangkaian dari *term* dengan frekuensi tertinggi dari masing – masing parameter.

Tabel 14. Jumlah Percobaan Sistem

Keyword	Banyak Kombinasi Percobaan
1 kata	269
2 kata	16
3 kata	30
4 kata	39
5 kata	23

4.3.1.2. Penentuan Treshold / Batasan *Similarity*

Sebelum masuk ke proses pengujian, terlebih dahulu ditentukan *threshold* yang akan menjadi batasan untuk mengembalikan nilai *similarity*. Nilai *threshold* ditentukan berdasarkan hasil percobaan dari pembentukan *keyword* yang sudah ditentukan. Nilai *threshold* diambil nilai *similarity* terendah dari percobaan yang sudah ditentukan. Nilai *threshold* juga dibagi berdasarkan jumlah kata pada *keyword*. Pada tabel 15 merupakan hasil penentuan nilai *threshold* berdasarkan jumlah kata percobaan untuk metode VSM dan *Weighted Tree Similarity*.

Tabel 15. Hasil Penentuan *Threshold*

Keyword	*Threshold*	
	VSM	*Weighted Tree Similarity*
1 kata	0.0747	0.0065
2 kata	0.1735	0.0562
3 kata	0.2999	0.0959
4 kata	0.7341	0.1906
5 kata	0.6625	0.1271

4.3.1.3. Analisa Hasil Pengujian Sistem

Setelah proses pengujian dari metode yang sudah ditentukan selesai, dilakukan analisis terhadap hasil pengujian dengan menghitung *precision* dan *recall* . Besarnya kecilnya nilai *precision* dan *recall* akan menunjukkan baik tidaknya sebuah metode. Kemudian juga pengujian terhadap waktu eksekusi juga dihitung untuk menghitung cepat tidaknya waktu eksekusi yang digunakan.

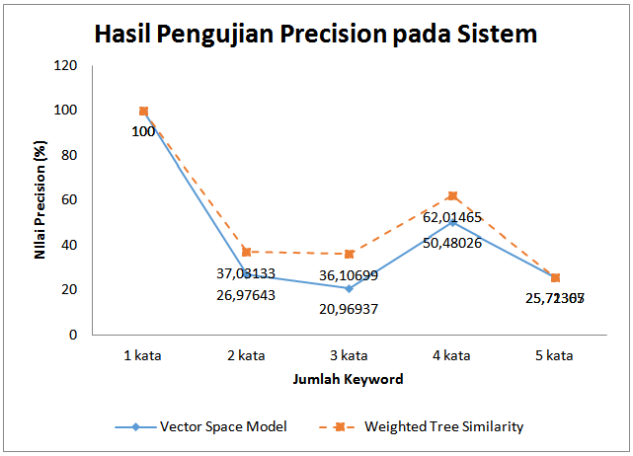
Tabel 16. Hasil Pengujian *Precision* dan *Recall* dengan Metode VSM

Keyword	*Precision*	*Recall*
1 kata	100 %	99,75609 %
2 kata	26,97643 %	100 %
3 kata	20,96937 %	100 %
4 kata	50,48026 %	100 %
5 kata	25,72307 %	95,65217 %
Rata - rata	**44,82983 %**	**99,08165 %**

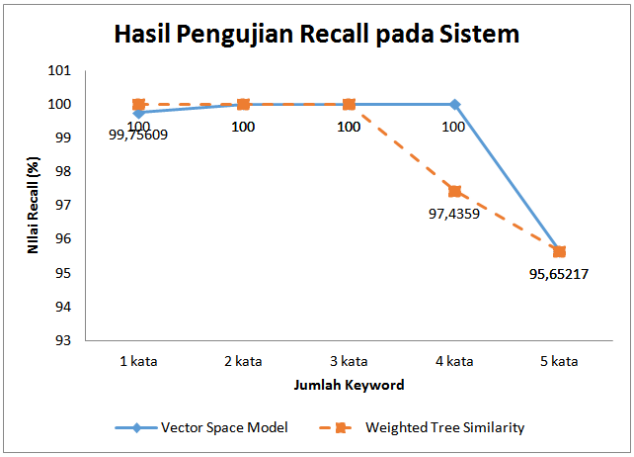
Tabel 17. Hasil Pengujian *Precision* dan *Recall* dengan Metode *Weighted Tree Similarity*

Keyword	*Precision*	*Recall*
1 kata	100 %	100 %
2 kata	37,03133 %	100 %
3 kata	36,10699 %	100 %
4 kata	62,01465 %	97,4359 %
5 kata	25,71365 %	95,65217 %
Rata - rata	**52,17332 %**	**98,61761 %**

Hasil pada percobaan dengan menggunakan metode VSM memberikan nilai rata- rata *precission* 44,82983 % dan *recall* 99,08165% yang bisa dilihat pada tabel 16. Hasil nilai *recall* sudah baik karena hampir mencapai 100%. Sedangkan pada percobaan dengan metode *Weighted Tree Similarity* memberikan hasil yang baik dengan nilai rata – rata *precision* 52,17332 dan *recall* 98,61761 bisa dilihat pada tabel 17. Dari rata – rata nilai *precision* metode ini lebih baik dibandingkan dengan metode VSM. Kemudian dari nilai *recall* sedikit lebih turun, namun selisihnya dengan metode VSM sangat kecil dan tidak terlalu signifikan.



Gambar 10. Grafik Pengujian *Precision* pada Sistem



Gambar 11. Grafik Pengujian *Recall* pada Sistem

Hasil pada percobaan dengan menggunakan metode VSM memberikan nilai rata- rata *precission* 44,82983 % dan *recall* 99,08165% yang bisa dilihat pada tabel 4.15. Hasil nilai *recall* sudah baik karena hampir mencapai 100%. Sedangkan pada percobaan dengan metode *Weighted Tree Similarity* memberikan hasil yang baik dengan nilai rata – rata *precision* 52,17332 dan *recall* 98,61761 bisa dilihat pada tabel 4.16. Dari rata – rata nilai *precision* metode ini lebih baik dibandingkan dengan metode VSM. Kemudian dari nilai *recall* sedikit lebih turun, namun selisihnya dengan metode VSM sangat kecil dan tidak terlalu signifikan.

Tabel 18. Hasil Pengujian Waktu dengan metode VSM

Keyword	**Waktu (detik)**	**Waktu *Hamming distance* (detik)**
1 kata	4,01223	1,31764
2 kata	6,54944	2,57761
3 kata	9,07661	3,82786
4 kata	12,32681	6,82902
5 kata	13,9621	8,00766
Rata - rata	**9,18544**	**4,51196**

Tabel 19. Hasil Pengujian Waktu dengan metode *Weighted Tree Similarity*

Keyword	**Waktu (detik)**	**Waktu *Hamming distance* (detik)**
1 kata	7,92163	1,5081
2 kata	10,88575	3,57722
3 kata	13,96942	5,25262
4 kata	18,8712	9,05895
5 kata	20,45582	10,81487
Rata - rata	**14,42076**	**6,04235**

Selanjutnya pada waktu eksekusi sistem dapat dilihat pada tabel 18 dan 19. Dari hasil tersebut diketahui bahwa pengaruh penggunaan algoritma *Hamming Distance* sangat signifikan. Penggunaan algoritma *Haming Distance* sangat membantu dalam mempercepat proses eksekusi. Karena dengan menggunakan algoritma ini, dokumen yang tidak memiliki kedekatan tidak akan dilanjutkan dalam perhitungan, sehingga waktu eksekusi menjadi lebih cepat.

4.3.2. Pengujian dengan Pakar

Pada pengujian dengan pakar dilakukan dengan menguji beberapa *keyword* yang sudah ditentukan. Selanjutnya *keyword* tersebut akan diuji dengan membandingkan metode VSM dan *Weighted Tree Similarity*. Peran pakar pada pengujian ini akan menentukan hasil pencarian mana saja yang relevan dan tidak relevan. Pengujian ini dilakukan dengan pakar yaitu Bapak dr. Burhanuddin Ichsan, M. Med. Ed, M.Kes. Beliau merupakan Dokter umum yang juga menjabat sebagai Dosen di UMS(Universitas Muhammadiyah Surakarta). Selain seorang dokter, penelitian yang sering dilakukan beliau juga terkait dengan penyakit umum dan kesehatan masyarakat.

Pengujian pakar dilakukan dengan 20 kali percobaan. Untuk hasil pengujian pakar yang asli dan hasil rekapitulasi pengujian pakar dapat dilihat pada lampiran A dan B. Setelah proses pengujian selesai, hasilnya diproses untuk dihitung nilai *precision* dan *recall.* Penghitungan *precision* dan *recall* dilakukan berdasarkan hasil pencarian yang dikembalikan dengan menunjukkan hasil relevan dan tidak relevan yang dipilih oleh pakar. Hasil pengujian yang sudah dilakukan bisa dilihat pada tabel 20 dan 21. Hasil *precision* lebih baik pada metode VSM, sedangkan nilai *recall* lebih tinggi pada metode *Weighted Tree Similarity*.

Tabel 20. Hasil Pengujian Pakar dengan metode VSM

Percobaan	**Precission**	***Recall***
20 kali	34.67365	85.69444

Tabel 21. Hasil Pengujian Pakar dengan metode *Weighted Tree Similarity*

Percobaan	**Precission**	***Recall***
20 kali	41.09162	71.66667

4.3.3. Analisis Metode Terbaik

Dari hasil pengujian pengujian sistem yang dilakukan, disimpulkan pada sub bab 4.4.1. *Weighted Tree Similarity* merupakan daripada metode VSM dengan memberikan nilai pengujian *precision* yang lebih baik. Hal ini juga didukung pada hasil *precision* pada pengujian pakar memberikan hasil lebih baik pada metode *Weighted Tree Similarity* dibandingan VSM*,* meskipun nilai rata – rata *recall* pada *Weighted Tree Similarity* lebih kecil dibandingkan VSM, selisihnya juga tidak terlalu jauh. Karena pada dasar pengujian *recall* dan *precision* metode pencarian yang lebih efektif adalah yang memberikan nilai ketepatan terbaik yaitu nilai *precision,* meskipun nilai *recall* lebih rendah.

Selanjutnya analisa hasil waktu menunjukkan metode VSM dengan *Hamming Distance* merupakan metode yang paling baik. Namun selisih waktu antara VSM dan *Hamming Distance* dengan *Weighted Tree Similarity* dan *Hamming Distance* tidak terlalu jauh, dan terpaut sekitar satu detik saja. Sehingga dari sini hasil tiga analisa pengujian yang dilakukan, dapat disimpulkan metode *Weighted Tree Similarity* dan *Hamming Distance* merupakan metode terbaik karena memberikan hasil yang lebih efisien dan efektif. Kemudian pada umumnya pencarian yang berhubungan dengan dunia kesehatan lebih mementingkan hasil pencarian yang efisien dan efektif.

5. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian dan pembahasan yang telah dilakukan, dapat disimpulkan bahwa metode *Weighted Tree Similarity* merupakan metode terbaik. Hal ini sudah dibuktikan pada pengujian sistem dan pakar, metode *Weighted Tree Similarity* memberikan nilai *precision* yang lebih baik dibandingkan dengan VSM, meskipun nilai *recall* lebih kecil dibandingkan pada metode VSM. Sedangkan perbedaan nilai *recall* juga tidak terlalu jauh.

Penggunaan algoritma Hamming Distance sangat berpengaruh dalam membantu mempercepat waktu eksekusi sistem. Hal ini dikarenakan pada algoritma Hamming Distance, apabila terdapat dokumen yang tidak memiliki kedekatan dengan *query* maka tidak akan dilanjutkan ke proses perhitungan selanjutnya.

Adapun beberapa saran yang dipertimbangkan untuk perkembangan penelitian selanjutnya yaitu :

1. Agar proses tidak terlalu lama, pada *term* yang memiliki frekuensi di semua dokumen dianggap sebagai *stopword*.
2. Supaya nilai *similarity* total mejadi lebih tinggi dan hasil yang diberikan lebih relevan, pada metode *Weighted Tree Similarity* dalam menghitung nilai *similarity* total diberikan nilai bobot masing – masing pada parameter yang lebih relevan. Pemilihan nilai bobot dengan melihat parameter mana yang dianggap lebih berpengaruh pada proses perhitungan *similarity*.
3. Untuk pencarian yang lebih relevan dan cepat, dapat mengoptimalkan penggunaan algoritma *Hamming Distance* dengan cara memberi batasan nilai *Hamming*. Batasan tersebut diberikan agar pencarian dengan kata kunci lebih dari satu kata dapat memberikan hasil yang sama dengan makna kata kunci yang dimasukkan tersebut.

6. DAFTAR PUSTAKA

[1] Rila, M., & Setiawan, H. (2001). *Improving Information Retrieval System Performance by Automatic Query Expansion*. *Makalah, Departemen Teknik Informatika, Institut Teknologi Bandung*.

[2] Zafikri A. (2010). Implementasi Metode *Term Frequency Inverse Document Frequency* (TF-IDF) pada Sistem Temu Kembali Informasi.

[3] Amin, F. (2012). Sistem Temu Kembali Informasi dengan Metode *Vector Space Model* . *Jurnal Sistem Informasi Bisnis 02* , 78-83.

[4] Perdana, P. A. (2014). Analisis Kombinasi Algoritma *Weighted Tree Similarity* Dengan Tanimoto Cosine (Tc) Untuk Pencarian Semantik Pada Portal Jurnal. *Prosiding SNST ke-5* , 79-84.

[5] Sarno, R., & Rahutomo, F. (2008). Penerapan Algoritma *Weighted Tree Similarity* Untuk Pencarian Semantik. *JUTI*, 35-42.

[6] Putro, L. S. (2014). Penerapan Kombinasi Algoritma Minhash dan *Binary Hamming Distance* pada *Hybrid* Perekomendasi Lagu.

[7] Sebastiani, F. 2002. *Machine Learning in Automated Text Categorization*

[8] Turney, P. D., & Pantel, P. (2010). From Frequency to Meaning: Vector Space . *Journal of Artificial Intelligence Research*, 141 -188.

[9] Robertson, Stephen 2005, *Understanding Inverse Document Frequency: On Theoretical Arguments for IDF*, *England Journal of Documentation*, Vol. 60, pp. 502 – 520.

[10] Susanto, S., & Sensuse, D. I. (2008, Vol.1 No.2). Pengklasifikasian Artikel Berita Berbahasa Indonesia secara Otomatis Menggunakan Naive Bayes Classifier. *Jurnal Ilmu Komputer dan Informasi*

[11] Sa'adah, Umi ; Sarno, Riyanarto ; Yuhana, Umi L. 2013. *Latent Semantic Analysis and Weighted Tree Similarity For Semantic Search In Digital Library*. The Proceedings of The 7th ICTS, Bali. 159 – 164.

[12] Clarke, S., & Willett, P. 1997. *Estimating the recall performance of search engines*. ASLIB Proceedings, 49 (7), 184-189.