Jurnal SimanteC
Vol. 4, No. 2 Desember 2014 ISSN 2088-2130

SISTEM FAQ KONSULTASI DOKTER GIGI MENGGUNAKAN ALGORITMA *WINNOWING* DAN *SYNONYM REPLACEMENT*

Wildan Faizzani[1)] , Fika Hastarita R.[2)]
[1]Prodi Teknik Informatika, Fakultas Teknik,Universitas Trunojoyo
Jl. Raya Telang, PO BOX 2 Kamal, Bangkalan
E-mail : [1]wildanfaizzani@gmail.com, [2]hastarita.fika@gmail.com

ABSTRAK

Kesehatan gigi amat penting bagi manusia, namun tidak semua orang masa kini memiliki waktu untuk pergi ke dokter gigi dalam rangka pencegahan penyakit gigi yang lebih kronis, akibatnya banyak orang mendatangi dokter gigi ketika keadaan giginya sudah rusak parah. Untuk menangani permasalahan ini, sangatlah tidak efisien untuk membangun aplikasi konsultasi konvensional. Mengingat jumlah pasien yang tidak sedikit dan kemungkinan terus bertambah kedepannya. Metode yang digunakan dalam penelitian ini adalah deteksi kemiripan dengan bantuan metode *Winnowing* dan didukung oleh *Synonym Replacement*. Dengan penyusunan FAQ dalam penerapannya menjadikan konsultasi akan semakin mudah. Uji coba dilakukan dengan menguji beberapa jenis *Synonym Replacement* dan diketahui melalui ujicoba *Synonym Replacement* Versi 4 yang terbaik untuk membantu pendeteksian dengan peningkatan angka deteksi sebesar 40% dibandingkan tanpa *Synonym Replacement*, serta dengan merubah konstanta algoritma *Winnowing* dan diketahui pula bahwa konstanta *prime* = 7, *n-gram* dan *n-window* = 3 adalah yang terbaik untuk deteksi kalimat, dengan angka deteksi mencapai 58%.

***Kata Kunci**:* Algoritma *Winnowing*, *Synonym Replacement*, dan *Frequently Ask Question*

ABSTRACT

Tooth's health was very important for human being, unfortunately nowadays people have no enough time to go to the dentist in order to prevents more cronical teeth problems, some of them went to the dentist when the condition was too late for prevention. At this rate, it is very not efficient to build the consulting application that require an answer manually. Considering the number of patient that keep grows by time. In this paper, we would represent our research in building an application such as the answer from problem stated above using Winnowing Algorithm and Synonym Replacement. This application using FAQ to compile and collect the same question in order to ease the process of consultation on the other hand.In this research number of tests are performed, in the test of several versions of Synonym Replacement found that Version 4 were the best in case similarity detection, compared to non-included Synonym Replacement similarity detection was approximately 40% higher, on the other side constant values of Winnowing tests found that prime = 7, n-gram and n-window = 3 constants were the best combination with similarity detection reaching 58%.

***Keywords**: Winnowing Algorithm, Synonym Replacement, and Frequently Ask Question*

PENDAHULUAN

Dalam setiap gigi dan gusi manusia terdapat ratusan bahkan lebih sistem syaraf dan pembuluh darah yang terhubung ke otak, jika terdapat masalah pada gigi dan gusi dibiarkan maka dikhawatirkan akan dapat menyebar kedalam sistem otak yang notabene sangat vital bagi manusia. Namun masih banyak diantara masyarakat yang belum memahami pentingnya menjaga kesehatan gigi dan mulut, kebanyakan dari mereka tidak memiliki waktu untuk berkonsultasi secara langsung.

FAQ (*Frequently Ask Question*) merupakan salah satu inovasi yang diperkenalkan oleh Eugene Miya pada tahun 1982 saat bertugas sebagai staf NASA untuk mengelola mailing list ARPAnet yang membahas masalah tentang ruang angkasa. Ide pembuatan FAQ berawal dari kesulitan yang dihadapinya untuk menjawab satu persatu pertanyaan *user* yang setiap hari terus bertambah[1].

Metode yang digunakan dalam penelitian ini adalah deteksi kemiripan dengan bantuan metode *Winnowing* dan didukung oleh *Synonym Replacement*.

Banyak cara atau metode yang dapat digunakan untuk mendeteksi kemiripan dalam file text. Namun ada kebutuhan mendasar yang harus dipenuhi oleh algoritma deteksi kemiripan, yaitu: (1.) *Whitespace Insensitivity* yang berarti dalam melakukan pencocokan terhadap file teks seharusnya tidak terpengaruh oleh spasi, jenis huruf (kapital atau normal), tanda baca dan sebagainya. (2.) *Noise Surpression* yang berarti menghindari penemuan kecocokan dengan panjang kata yang terlalu kecil atau kurang relevan, misal: ‘the’. Panjang kata yang ditengarai merupakan suatu kemiripan harus cukup untuk membuktikan bahwa kata-kata tersebut benar-benar memiliki kemiripan dan bukan merupakan kata yang umum digunakan. (3.) *Position Independence* yang berarti penemuan kemiripan harus tidak bergantung pada posisi kata-kata. Meskipun posisinya tidak sama, kemiripan harus dapat ditemukan[2].

METODE

Preprocessing

Tugas pokok dari *preprocessing* adalah membuat *Index* dari kumpulan dokumen. *Index* dokumen adalah himpunan *term* yang menunjukkan isi atau topik yang dikandung oleh dokumen. *Index* berfungsi untuk membedakan suatu dokumen dari dokumen lain. Ukuran *index* sangat mempengaruhi dari hasil yang akan ditampilkan. Semakin kecil ukuran *index* maka semakin kecil pula relevansi antar dokumen yang akan didapat. Semakin besar ukuran *index* maka akan semakin besar pula relevansi tiap dokumen namun sekaligus menaikkan jumlah dokumen yang tidak relevan dan menurunkan kecepatan pencarian. [3]

Terdapat 4 dari 5 langkah pembangunan *inverted index*, yaitu [3]:

1. Penghapusan format dan markup dari dalam dokumen

   Tahap ini menghapus semua tag *markup* dan format khusus dari dokumen, terutama pada dokumen yang mempunyai banyak *tag* dan format seperti dokumen (X)HTML. Jika isi dokumen telah berada di dalam *database* maka tahapan ini sering ditiadakan.
2. Pemisahan rangkaian *term* (*tokenization*)

   Tokenization adalah tugas memisahkan deretan kata di dalam kalimat, paragraf atau halaman menjadi *token* atau potongan kata tunggal atau *termed word*. Tahapan ini juga menghilangkan karakter-karakter tertentu seperti tanda baca dan mengubah semua *token* ke bentuk huruf kecil (*lower case*).
3. Penyaringan (*filtration*)

   Pada tahapan ini ditentukan *term* mana yang akan digunakan untuk merepresentasikan dokumen sehingga dapat mendeskripsikan isi dokumen dan membedakan dokumen tersebut dari dokumen lain di dalam koleksi. *Term* yang sering dipakai tidak dapat digunakan untuk tujuan ini, setidaknya

karena dua hal. Pertama, jumlah dokumen yang relevan terhadap suatu *query* kemungkinan besar merupakan bagian kecil dari koleksi. *Term* yang efektif dalam pemisahan dokumen yang relevan dari tidak relevan kemungkinan besar adalah *term* yang muncul pada sedikit dokumen. Ini berarti bahwa *term* dengan frekuensi tinggi merupakan *poor descriminator*. Kedua, *term* yang muncul dalam banyak dokumen tidak mencerminkan definisi dari topik atau sub-topik dokumen. Karena itu, *term* yang sering digunakan dianggap sebagai *stopword* dan dihapus.

4. Pengembalian *term* ke bentuk akar kata (*stemming*)

Kata-kata yang muncul di dalam dokumen sering mempunyai banyak varian morfologik. Karena itu, setiap kata yang bukan *stopwords* direduksi ke *stemmed word* (*term*) yang cocok yaitu kata tersebut di*stem* untuk mendapatkan bentuk akarnya dengan menghilangkan awalan atau akhiran [4].

ECSP Stemmer

Adapun langkah-langkah penyusunan algoritma *stemming* ECSP adalah sebagai berikut :

1. Melakukan pengecekan jumlah karakter/huruf dalam kata *input*-an, jika kata yang akan di *stem* mempunyai jumlah karakter/huruf < 6, maka kata tersebut tergolong tidak berimbuhan dan secara langsung akan di *return* oleh *Stemmer*. Algoritma ini disebut cek *non affiks*.
2. Pengecekan *ilegal affiks* (kata imbuhan yang tidak diperbolehkan). Contoh *ilegal affiks* pada aturan Bahasa Indonesia yaitu:ke-..-i|-kan, se-..-i|kan, peng-..-..-i|kan, tar-..-an.
3. Menghapus kata ganti kepunyaan yang berada di depan seperti: ku-, kau-, dan serapan awalan asing seperti : adi-,antar-,anti-,anu-,audio-,awa-,bi-,catur-,dasa-,deka-,dwi-,eka-,infra-,maha-,manca-,multi-,nara-,pasca-,pari-,pramu-,pra-,sapta-,semi-,swa-,tri-,ultra-.
4. Menghapus *inflectional particle* P (-lah, -kah, -tah, -pun) dan kata ganti kepunyaan atau *possessive pronoun* PP (-ku, -mu, -nya).
5. Menghapus awalan .
   - Menghapus awalan yang tidak bermofologi seperti (di-, ke-, se-).
   - Menghapus awalah bermofologi (be-, te-, pe-, me-).

Lakukan *recording* sesuai dengan tabel pemenggalan imbuhan ECSP dan yang sudah dikembangkan dari aturan pemenggalan awalan *ECS* dan *Porter Stemmer*.[5]

Synonym Replacement

Untuk pendeteksian kalimat sama, jenis kata sangat berpengaruh pada sensitifitas deteksi. *Synonym Replacement* membantu memecahkan permasalahan ketika terdapat dua kata yang memiliki makna sama namun dengan tulisan yang berbeda. Perlu diketahui bahwa metode yang digunakan tidak menggunakan analisa leksikal dalam penerapannya.

Terdapat empat versi modul *Synonym Replacement* yang dikembangkan dalam penelitian ini. Versi 1 (selanjutnya disebut V.1) menggantikan kata yang di deteksi sama dengan sinonim urutan pertama dalam kamus. Versi 2 (V.2) menggabungkan sinonim urutan pertama dalam kamus tepat dibelakang kata yang dituju. Versi 3 (V.3) menambahkan salah satu sinonim yang tersimpan ke belakang susunan kata tanpa merubah susunan kata didepannya. Versi 4 (V.4) menambahkan semua sinonim yang dikenali dalam kamus dibelakang urutan susunan kata yang dimaksud.

Algoritma *Winnowing*

Berikut contoh implementasi algoritma *Winnowing* dalam melakukan proses *document fingerprinting* pada teks "Kenapa gigi pada anak belum tumbuh seusianya?":

(1) Melakukan proses *whitespace insensitivity*, sehingga hal yang mengandung huruf kapital dijadikan *ignore case*, tanda baca, spasi, dan

karakter-karakter yang tidak relevan lainnya dibuang. Sehingga dari kalimatnya diubah menjadi:
kenapagigipadaanakbelumtumbuhseusianya
(2) Setelah kalimat tersebut dibersihkan, pembentukan rangkaian *gram* dengan ukuran 4-*gram* menjadi:
kena enap napa apag pagi agig gigi igip gipa ipad pada adaa daan aana anak nakb akbe kbel belu elum lumt umtu mtum tumb umbu mbuh buhs uhse hseu seus eusi usia sian iany anya
(3) Penghitungan nilai-nilai *hash* dari setiap *gram* (sebuah hipotesis nilai *hash* yang muncul):
155945 148920 159476 143829 162047 142828 151036 153485 151127 154474 162006 142371 146014 142151 143591 159422 143233 155494 143964 148895 159220 170309 160511 169850 170111 158328 145854 169677 153567 166688 149958 170894 166947 152823 143845
(4) Untuk memilih hasil yang telah di *hash*, dilakukan dengan membagi ke *window* w dengan panjang 4. Kemudian pilih nilai yang minimum.
[155945 148920 159476 143829]
[148920 159476 143829 162047]
[159476 143829 162047 142828]
[143829 162047 142828 151036]
[162047 142828 151036 153485]
[142828 151036 153485 151127]
[151036 153485 151127 154474]
[153485 151127 154474 162006]
[151127 154474 162006 142371]
[154474 162006 142371 146014]
[162006 142371 146014 142151]
[142371 146014 142151 143591]
[146014 142151 143591 159422]
[142151 143591 159422 143233]
[143591 159422 143233 155494]
[159422 143233 155494 143964]
[143233 155494 143964 148895]
[155494 143964 148895 159220]
[143964 148895 159220 170309]
[148895 159220 170309 160511]
[159220 170309 160511 169850]
[170309 160511 169850 170111]
[160511 169850 170111 158328]
[169850 170111 158328 145854]
[170111 158328 145854 169677]
[158328 145854 169677 153567]
[145854 169677 153567 166688]
[169677 153567 166688 149958]
[153567 166688 149958 170894]
[166688 149958 170894 166947]
[149958 170894 166947 152823]
[170894 166947 152823 143845]
(5) Setelah itu memilih nilai *hash* yang paling minimum yang telah dibagi menjadi *window* dengan urutan nilai *index array* secara berkelanjutan. Berikut dengan penambahan informasi posisi *fingerprint* di dalam dokumen. Hasilnya adalah sebagai berikut:
[143829,3] [142828,5] [151036,6] [151127,8] [142371,11] [142151,13] [143233,16] [143964,18] [148895,19] [159220,20] [160511,22] [158328,25] [145854,26] [149958,30] [143845,34]

Hashing

Hashing adalah suatu cara untuk mentransformasi sebuah *string* menjadi suatu nilai yang unik dengan panjang tertentu (*fixed-length*) yang berfungsi sebagai penanda *string* tersebut. Fungsi untuk menghasilkan nilai ini disebut fungsi *hash*, sedangkan nilai yang dihasilkan disebut nilai *hash*.
Contoh sederhana *hashing* adalah:
Amir, Budi
Marcell, Hendras
Gugun, Dimas
Maryam, Fitria
Menjadi
143565 = Amir, 145800 = Budi
210536456 = Marcell, 202267081 = Hendras
1677610 = Gugun, 1618226 = Dimas
19142287 = Maryam, 18133949 = Fitria

Contoh di atas adalah penggunaan *hashing* dalam pencarian pada *database*. Apabila tidak di-*hash*, pencarian akan dilakukan karakter-per-karakter pada nama-nama yang panjangnya bervariasi dan ada 26 kemungkinan pada setiap karakter. Namun pencarian akan menjadi lebih efisien setelah di-*hash* karena kemungkinan setiap angka berbeda. Nilai *hash* pada umumnya digambarkan sebagai *fingerprint* yaitu *string* pendek yang terdiri atas huruf dan angka yang

terlihat acak (data biner yang ditulis dalam heksadesimal)[2].

FAQ

FAQ merupakan inovasi dari perkembangan teknologi informasi dan komunikasi yang berisi dokumen informasi tentang jawaban terhadap suatu pertanyaan yang sering dilontarkan oleh pengguna pada sebuah *newsgroup* Penggunaan FAQ ini diperkenalkan oleh Eugene Miya pada tahun 1982 saat bertugas sebagai staf NASA untuk mengelola *mailing list* ARPAnet yang membahas masalah tentang ruang angkasa.

Ide pembuatan FAQ berawal dari kesulitan yang dihadapinya untuk menjawab satu persatu pertanyaan *user* yang setiap hari terus bertambah. Untuk mengatasinya, seluruh pertanyaan yang sering ditanyakan oleh *user* pada *mailing list* disusun dalam sebuah *database*, kemudian secara rutin dilakukan update pertanyaan dan jawaban pada FAQ.[1]

Tidak ada peraturan siapa yang boleh/tidak untuk menyusun/menulis FAQ, selama ada kebutuhan dari informasi yang terkandung dalam FAQ yang dibuat, FAQ anda akan diapresiasi[7].

ALGORITMA *WINNOWING* DAN *SYNONYM REPLACEMENT* DALAM PENERAPANNYA

Untuk dapat menemukan identitas *fingerprint* pada teks, dalam penelitian dilakukan prosedur-prosedur berikut ini.

(1) *Useless Character Filter*

Menghilangkan karakter-karakter yang tidak dipakai dalam deteksi seperti tanda baca.

Input : apakah benar kafein menunda rasa kantuk?

Output : apakah benar kafein menunda rasa kantuk

(2) *Text Splitter*

Memecah suatu kalimat menjadi *array* kata

[0 => apakah] [1 => benar] [2 => kafein] [3 => menunda] [4 => rasa] [5 => kantuk]

(3) *Stopword Filter*

Menghilangkan kata hubung, kata tanya, kata ganti, dan kata-kata lainnya yang dianggap tidak membantu dalam deteksi.

[0 => kafein] [1 => menunda] [2 => kantuk]

(4) *Synonym Replacement*

Menambahkan semua padanan kata yang terdapat dalam *database* kedalam urutan terakhir dalam *array* kata

[0 => kafein] [1 => menunda] [2 => kantuk] [3 => kemudian] [4 => membatalkan] [5 => memperlalaikan] [6 => mendorong] [7 => mengundurkan] [8 => menjorokkan]

(5) ECSP *Stemmer*

Merubah kalimat berimbuhan menjadi kalimat dasar

[0 => kafein] [1 => tunda] [2 => kantuk] [3 => mudi] [4 => batal] [5 => lalai] [6 => dorong] [7 => kundur] [8 => jorok]

(6) *Stopword Filter*

[0 => kafein] [1 => tunda] [2 => kantuk] [3 => mudi] [4 => batal] [5 => lalai] [6 => dorong] [7 => kundur] [8 => jorok]

(7) *Whitespace Filter*

Menggabungkan kembali *array* kata menjadi satu kalimat tanpa spasi

kafeintundakantukmudibatallalaidorongkundurjorok

(8) *n-Gram*

membentuk rangkaian *n-gram* dari teks [2], semisal n=7

kafeint afeintu feintun eintund intunda ntundak tundaka undakan ndakant …. ngkundu gkundur kundurj undurjo ndurjor durjoro urjorok

untuk teks tersebut dihasilkan 41 *gram*

(9) *Hashing*

Melakukan fungsi *hash* pada setiap *n-gram*[2]

113954 107970 111881 112669 117217 122123 125896 124106 116555.... 118452 114900 119545 124737 118446 114879 126044

Fungsi *hash* H(c1...ck) didefinisikan sebagai berikut:

$$c_1 * b^{(k-1)} + c_2 * b^{(k-2)} * \ldots + c_k - 1 * b + c_k \quad (1)$$

Keterangan:
c: nilai *ascii* karakter
b: basis (bilangan prima)
k: banyak karakter

Sebagai contoh n-gram dari "*kafeint*" dengan $b = 3$ dan $k = 7$
$H_{(kafeint)} = ascii(k)*3^{(6)} + ascii(a)*3^{(5)} + ascii(f)*3^{(4)} + ascii(e)*3^{(3)} + ascii(i)*3^{(2)} + ascii(n)*3^{(1)} + ascii(t)*3^{(0)}$
$H_{(kafeint)} = 107*729 + 97*243 + 102*81 + 101*27 + 105*9 + 110*3 + 116*1$
$H_{(kafeint)} = 113954$
(10) *n-Window*

Adalah proses pemecahan urutan *hash* dari *n-gram* menjadi *n-window* dengan *n* yang dapat tidak bergantung pada nilai *n* sebelumnya.

Dalam contoh ini misal *n-window* dengan nilai *n* = 7
113954 **107970** 111881 112669 117217 122123 125896
107970 111881 112669 117217 122123 125896 124106
111881 112669 117217 122123 125896 124106 116555
…..
118452 114900 119545 124737 118446 **114879** 126044

Dihasilkan 35 n-*window*.

dan masing masing *window* akan dipilih nilai hash terkecil.
(11) *Fingerprint*

Didapat *fingerprint* dari contoh diatas.
107970 108421 108794 108996 109043 109212 110415 111405 111881 113510 114879 114900

Nilai terkecil dari masing-masing *n-window* akan diseleksi dan digunakan sekali saja, nilai terkecil yang sama pada masing-masing *window* tidak membuat nilai *fingerprint* menjadi homogen, karena setiap anggota nilai *hash*-nya bernilai unik.

HASIL DAN PEMBAHASAN

Uji Coba sistem keseluruhan dilakukan dengan menentukan *query* yang akan diujikan secara acak namun mengarah kepada materi tertentu. Berdasarkan dokumen materi yang telah didapat yaitu sebanyak 64 butir, maka jumlah skenario yang di uji coba tidak lebih dari 10 butir saja. Setiap *query* uji coba akan diulang sebanyak 10 kali untuk menguji kelayakan sistem FAQ.

Gambar 1. *Interface* Untuk Konsultasi - *Input*

Gambar 2. *Interface* Untuk Konsultasi – *Output*

Tabel 1. Daftar Skenario Uji Coba Sistem

No	Skenario Pertanyaan
1	Apabila gigi anak goyang perlu dicabut?
2	Nyeri pada pipi
3	Mulut tidak bisa menutup kembali
4	Gigi terasa ngilu bila terkena angin
5	Sakit yang menyengat pada malam hari
6	Makanan yang menekan kerusakan gigi
7	Mulut kering dan bau mulut
8	Berdarah saat menyikat gigi
9	Gigi tidak beraturan
10	Nyeri pasca pencabutan gigi

Tabel 2. Daftar Jawaban Yang di Dapat

No	Jawaban
1	Jawaban *Default*
2	Jawaban *Default*
3	karena adanya tulang rahang yang bergeser (disposisi) sehingga mengunci tidak bisa menutup kembali
4	pembuatan gangiva tiruan, bila terdapat resesi pada gingiva dan pemeriksaan radiologis terdapat kerusakan tulang alveolaris
5	Jawaban *Default*
6	makanan yang berserat tinggi
7	biasanya terdapat penyakit seperti diabetes yang mendukung kelainan ini, senyawa

	yang menyebabkan bau mulut adalah aseton yang kadarnya meningkat dalam darah pasien
8	Jawaban *Default*
9	Jawaban *Default*
10	Jawaban *Default*

Tabel 3. Daftar FAQ Konsultasi Yang di Dapat

No	Skenario Pertanyaan
1	Apabila gigi anak goyang perlu dicabut?
2	Nyeri pada pipi
3	Sakit yang menyengat pada malam hari
4	Berdarah saat menyikat gigi
5	Gigi tidak beraturan
6	Nyeri pasca pencabutan gigi

Tabel 4. Daftar FAQ Materi Yang di Dapat

No	Q	A
1	setelah menguap, tidak bisa menutup mulut kembali, apakah penyebabnya?	karena adanya tulang rahang yang bergeser (disposisi) sehingga mengunci tidak bisa menutup kembali
2	mulut kering sejak tiga bulan lalu, beberapa gigi goyang, dan bau mulut, apa yang terjadi?	biasanya terdapat penyakit seperti diabetes yang mendukung kelainan ini, senyawa yang menyebabkan bau mulut adalah aseton yang kadarnya meningkat dalam darah pasien
3	makanan apa yang bisa menekan resiko kerusakan gigi?	makanan yang berserat tinggi
4	gigi rahang depan terasa memanjang dan linu bila terkena angin. perawatan apa yang seharusnya didapat pasien?	pembuatan gangiva tiruan, bila terdapat resesi pada gingiva dan pemeriksaan radiologis terdapat kerusakan tulang alveolaris

Tabel 5. Hasil Uji Coba *Stopword Filter*

No.	Posisi S. *Filter* dan Deteksinya (%)						
Skn.	1	2	3	1&2	2&3	3&1	1,2&3
1	50	36	37	50	37	50	50
2	24	36	46	24	46	38	38
3	79	53	79	79	80	81	81
4	65	37	53	63	57	65	65
5	43	32	40	43	40	43	43
6	96	50	93	96	93	96	96
7	79	56	59	78	63	78	78
8	45	33	42	45	42	45	45
9	9	16	29	9	29	9	9
10	43	27	31	43	37	43	43
Avg.	53	38	51	53	52	55	55

Tabel 6. Hasil Uji Coba *Synonym Replacement*

No.	Versi S. *Replacement* dan Deteksinya (%)				
Skn.	Tanpa	V.1	V.2	V.3	V.4
1	36	33	33	29	50
2	33	33	33	23	38
3	29	29	38	33	81
4	35	39	41	41	65
5	26	15	34	25	43
6	58	64	75	74	96
7	58	56	65	65	78
8	25	26	31	29	45
9	25	22	15	18	9
10	25	43	46	28	43
Avg.	35	36	41	37	55

Tabel 7. Hasil Uji Coba *Winnowing*

No. Skn.	Kombinasi Konstanta (Prime-Gram-Wind) dan Deteksinya (%)					
	11-4-3	7-4-3	11-3-4	7-3-4	11-3-3	7-3-3
1	45	45	43	42	50	50
2	35	35	38	38	38	38
3	78	77	79	78	81	81
4	62	63	62	62	63	65
5	40	41	44	45	43	43
6	94	94	96	96	95	96
7	70	70	72	73	77	78
8	41	41	46	45	46	45
9	4	4	11	11	9	9
10	32	31	43	41	43	43
Avg.	50	50	53	53	54	55

Tabel 8. Hasil Uji Coba Threshold

No. Skn.	Jawaban Diinginkan (ID_MATERI)	Jawaban Diperoleh	
		Kesamaan (%)	ID_MATERI
1	8	50	8
2	43	38	35
3	33	81	33
4	60	65	60
5	30	43	30
6	54	96	54
7	46	78	46
8	56	58	55
9	7	20	1
10	20	43	20

Pada Tabel 2 dapat dilihat bahwa skenario uji coba nomor 3, 4, 6 dan 7 menghasilkan jawaban. Pada Tabel 4 merupakan kumpulan F.A.Q. Materi dari 4 skenario yang menghasilkan jawaban. Sedangkan skenario uji coba nomor 1, 2, 5, 8, 9 dan 10 yang tidak menghasilkan jawaban (*default*) menjadi F.A.Q Konsultasi yang memungkinkan admin untuk menjawabnya ada pada Tabel 3. Secara keseluruhan sistem berjalan baik, sistem FAQ untuk dokumen Materi maupun dokumen pertanyaan juga berjalan dengan baik.

Pada Tabel 5 terlihat bahwa kombinasi posisi *Stopword Filter* 3 *and* 1 menempati urutan teratas bersamaan dengan posisi 1, 2 *and* 3. Karena proses yang digunakan oleh 3 *and* 1 lebih sedikit, maka posisi inilah yang dianggap paling efektif. Untuk uji level individual, posisi 1 menempati urutan teratas

berdasarkan nilai rata-rata pada uji coba ini, hal ini dikarenakan *stopword filter* dilakukan di awal pemecahan kalimat. Namun pada level ini, kelemahannya adalah ketika *stemming* dilakukan, masih terdapat kemungkinan adanya *stopword*. Urutan dibawahnya adalah posisi 3 dimana *stopword filter* dilakukan setelah *stemmer*, yaitu tingkat akhir preproses teks sebelum digabungkan seluruhnya, namun kelemahannya adalah tidak dilakukannya *filter* awal yang dapat menyebabkan banyaknya *stopword* yang kemungkinan memiliki banyak sinonim yang dapat membuat hasil lebih divergen.

Pada Tabel 6 *Synonym* V.4 adalah yang paling efektif untuk menemukan jawaban yang diinginkan. Dalam Tabel 8 terlihat bahwa V.1 tidak banyak membantu dalam meningkatkan deteksi kemiripan, hal ini dikarenakan V.1 hanya mengganti kalimat yang dimaksud dengan salah satu sinonim dari sekian banyaknya *database*. Sedangkan V.4 menempati urutan teratas dalam membantu meningkatkan sensitifitas kesamaan, karena V.4 memasukkan semua sinonim yang memiliki arti sama kedalam *array* kata yang diproses, ini juga dapat memecahkan permasalahan dari algoritma *winnowing* pada sistem ini karena keterbatasan panjang teks yang tersedia.

Tabel 7 Menunjukkan bahwa kombinasi terbaik dalam uji coba ini adalah Bilangan Prima = 7, *n-Gram* = 3, dan *n-Window* = 3. Berdasarkan Tabel 7 dapat dilihat bahwa yang paling berpengaruh terhadap sensitifitas deteksi kemiripan adalah konstanta n pada *n-Gram*, disusul oleh n pada *n-Window* dan yang terakhir adalah Bilangan Prima pada *Hashing*. Berdasarkan hasil yang ditunjukkan oleh Tabel 8, *Threshold* yang digunakan adalah rata-rata dari nilai kesamaan skenario uji coba yang memiliki jawaban cocok dengan prakiraan jawaban sebelumnya. **(50% + 81% + 65% + 43% + 96% + 78% + 43%) / 7 = 65%.** Setelah pembulatan bilangan, *threshold* yang digunakan adalah 65%. Bilangan *threshold* yang tinggi dapat mencegah kesalahan deteksi dalam pencariannya di dokumen materi.

SIMPULAN

Algoritma *Winnowing* dapat membantu mendeteksi kemiripan antara dua teks meskipun memiliki kelemahan yaitu panjang kalimat yang terbatas dapat mengurangi sensitifitas deteksi. Namun, *Synonym Replacement* dapat menutupi kelemahan dari kondisi ini dengan menambahkan semua sinonim yang ada dalam database jika salah satu kata yang diproses terdeteksi terdapat kata induk sinonim. Sedangkan F.A.Q. dapat memudahkan baik *admin* maupun *user* dalam memonitor pertanyaan yang sudah sering ditanyakan.

DAFTAR PUSTAKA

[1] Daulay, P., dan Badrus Z. *Pengembangan Model Penelusuran Diskusi Tutorial Online Melalui Aplikasi FAQ (Frequently Ask Question)*. UPBJJ-UT Surabaya. 2012.

[2] Purwitasari, D., Putu Yuwono K., dan Umi Laili Y. *Deteksi Keberadaan Kalimat Sama Sebagai Indikasi Penjiplakan Dengan Algoritma Hashing Berbasis N-Gram*. Jurnal Ilmiah KURSOR. Surabaya.2011.

[3] Mastur,M. *Perbandingan efektifitas antara penghapusan stoplist dengan penghapusan stoplist dan kata umum pada dokumen hasil klasifikasi pretopology*. Bangkalan: Skripsi Jurusan Teknik Informatika Fakultas Teknik Universitas Trunojoyo Madura; 2012.

[4] Manning, Christopher, D., Raghavan, P., dan Schütze ,H. *An Introduction to Information retrieval*. Cambridge: University Press; 2008.

[5] Fahmi,A. *Rancang Bangun Sistem Pencarian Dan Hirarki Pasal-Pasal Tentang Lalu Lintas Angkutan jalan Dengan Menggunakan Vector Space Model*. Bangkalan: Skripsi Jurusan Teknik Informatika Fakultas Teknik Universitas Trunojoyo Madura. 2013.

[6] Pratama, Mudafiq R., Eko Budi C., dan Gita Indah M. *Aplikasi Pendeteksi Duplikasi Dokumen Teks Bahasa Indonesia Menggunakan Algoritma Winnowing Dengan Metode K-Gram Dan Synonym Recognition*. Teknik Informatika UM Malang. 2012.

[7] Hersh, R**.** 1995. FAQ*s about* FAQ*s*. URL: http://en.wikipedia.org/wiki/FAQ. diakses 21 Januari 2013.