EVALUASI EFEKTIVITAS PENCARIAN DOKUMEN HTML PADA PENERAPAN *STEMMED TERM VECTOR* MODEL DENGAN PEMBOBOTAN LOGARITMA FREKUENSI *TERM* DALAM DOKUMEN DAN PENGUKURAN SOKAL

Angga Kusuma Nugraha[1)], **Yesi Puspita Dewi** [2)]
[1]Teknik Informatika, Fakultas Teknologi Informasi, Universitas Budi Luhur
[2]Sistem Informasi, Fakultas Teknologi Informasi, Universitas Budi Luhur
Jl. Raya Ciledug, Pertukangan Utara, Jakarta Selatan 12260
Telp : (021)5853489
E-mail : incredibleangga@gmail.com[1)], yesipuspita.dewi@gmail.com[2)]

Abstract

Purpose of this study was to evaluate the effectiveness of an information retrieval system using the sokal / sneath equation and vector method by comparing between one system model and another. Number of documents used 100 documents taken from the internet with 10 different topics. Algorithms used include algorithm for extact, tokenization, stopword removal, stemming, term weighting and similarity calculation. An information acquisition system is a system that automatically searches for information that is relevant to user needs. Finally, in measuring the relevance of the document to the query is done using the calculation of the average precision and recall. The calculations are done manually. From the calculation of the average precision and recall will get a collection point of effectiveness later on at the end of this study will be described in graphical form. Furthermore, the graph is incorporated into several graphs of precision and recall in other related studies, for comparison in obtaining final conclusions..

Keywords: *Information retrieval, Precision and recall, Similarity Calculation, Stemming, Sokal, Stopword removal, Tokenization*

Abstrak

Tujuan penelitian ini adalah untuk mengevaluasi efektivitas dari suatu sistem perolehan informasi (*information retrieval*) yang menggunakan ukuran kesamaan sokal / sneath dan metode vektor dengan cara membandingkan antara satu model sistem dengan model yang lain. Jumlah dokumen yang digunakan 100 dokumen yang diambil dari internet dengan 10 topik berbeda. Algoritma yang digunakan antara lain algoritma untuk *ekstact*, *tokenization*, *stopword removal*, *stemming*, pembobotan (*term weighting*) dan perhitungan *similarity*. Sistem perolehan informasi adalah sistem yang secara otomatis melakukan pencarian untuk memperolehan informasi yang relevan terhadap kebutuhan pengguna. Pada akhirnya nanti, dalam pengukuran relevansi dokumen terhadap query dilakukan dengan menggunakan perhitungan rata-rata *precision* dan *recall*. Perhitungannya dilakukan dengan cara manual. Dari hasil perhitungan rata-rata *precision* dan *recall* akan didapatkan kumpulan titik efektifitas yang nanti pada akhir penelitian ini akan digambarkan dalam bentuk grafik. Selanjutnya grafik tersebut disatukan dengan beberapa grafik rata-rata *precision* dan *recall* pada penelitian yang terkait lainnya, untuk dilakukan perbandingan dalam memperoleh kesimpulan akhir.

Kata Kunci: *Information retrieval, Precision dan recall, Similarity Calculation, Stemming, Sokal, Stopword removal, Tokenization*

1. PENDAHULUAN

Sebuah sistem perolehan informasi digunakan untuk mengurangi jumlah informasi yang berlebih, agar pencari informasi dapat lebih mudah mendapatkan informasi yang diinginkannya. Sistem perolehan informasi web yang saat ini dikenal oleh banyak orang adalah *search engine* atau mesin pencari.

Cara mesin pencari menentukan halaman mana yang paling sesuai, dan urutan halaman-halaman itu diperlihatkan, sangat bervariasi. Metodenya pencariannya berubah seiring waktu dengan berubahnya penggunaan internet dan berevolusinya teknik-teknik baru. Karena banyak metode telah ditemukan dan metode-metode tersebut punya kelebihan dan kekurangannya masing-masing, maka timbul permasalahan untuk mencari metode yang paling efektif dalam upaya pencarian dokumen.

Dalam penelitian ini menggunakan sistem perolehan informasi web, dimana fokus pencarian adalah informasi yang terdapat dalam dokumen yang terdapat pada web. Metode yang digunakan adalah metode pembobotan berdasarkan pembagian antara frekuensi *term* dalam satu dokumen dengan jumlah seluruh frekuensi *term* dalam dokumen tersebut dan dengan menggunakan metode pengukuran kesamaan sokal.

1.1 Tujuan Penelitian

Penelitian ini pada dasarnya bertujuan untuk menemukan teknik kombinasi ataupun menemukan teknik perolehan informasi yang tepat, yang dapat meningkatkan relevansi atau kesesuaian yang diperoleh dengan keinginan pencari informasi pada koleksi dokumen.

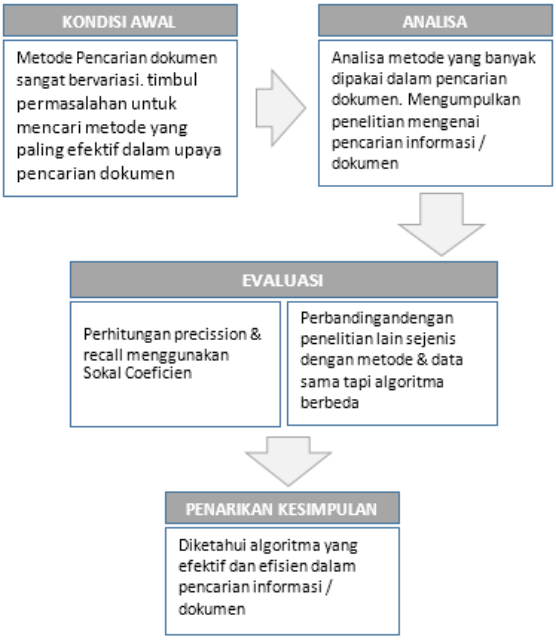
1.2 Manfaat Penelitian

Peneliti dapat merinci manfaat penelitian adalah sebagai berikut:

a. Untuk mengetahui cara penerapan metode pembobotan berdasarkan pembagian jumlah frekuensi kemunculan suatu term dengan jumlah frekuensi kemunculan suatu term pada database sebagai alternatif pencarian data, untuk mendapatkan hasil pencarian yang tepat dan akurat.
b. Untuk mengetahui cara kerja dari model pendekatan pengukuran kesamaan Sokal.
c. Mempersingkat dan mempercepat waktu untuk mencari informasi yang diinginkan, serta memudahkan dalam mendapatkan data atau informasi.
d. Membantu user dalam banyak hal seperti : pencarian dokumen, dan informasi yang dibutuhkan berdasarkan keyword (query)

1.3 Kerangka Pemikiran

Pendekatan untuk solusi mengatasi masalah yang ada dengan menggunakan diagram yang dapat dilihat pada gambar berikut.



Gambar 1 : Kerangka Pemikirian

2. METODE

Metode riset yang tepat dan benar semakin diperlukan dan menjadi sangat penting bagi keberhasilan suatu riset (penelitian). Salah satu hal yang penting dalam setiap penelitian adalah membuat rumusan metodologi penelitian. Melalui metodologi harus dengan jelas tergambar diantaranya bagaimana cara penelitian dilaksanakan yang tertata secara sistimatis; bagaimana landasan teori tentang rancangan penelitian (*research design*), model yang digunakan (didahului dengan rancangan percobaan (penelitian eksperimen)) atau teknik-teknik yang biasa digunakan dalam pengumpulan, pengolahan dan analisa data.

2.1 Konsep Pencarian Dokumen

Konsep pencarian dokumen pada penelitian ini adalah dengan mencocokan query dari pengguna dengan dokumen yang tersimpan pada database sistem. Sebelum query dicocokkan dengan *document term* yang terdapat dalam database, dokumen-dokumen yang ada akan mangalami proses *ekstract* atau *parsing* dokumen kedalam bentuk teks, hal ini dilakukan agar dokumen dengan format HTML tidak lagi menyertakan *tag* HTML yang akan mempengaruhi atau memperlambat proses perolehan *term*. Setiap dokumen yang mengandung *term* akan mengalami proses *stemming* sehingga dapat mengurangi variasi kata yang terdapat dalam dokumen.

Pada penelitian ini konsep pencarian dokumen dilakukan dengan menggunakan metode pembobotan berdasarkan logaritma frekuensi *term* dalam satu dokumen dan *sokal similarity measurement* sebagai pengukur kesamaan dokumen dengan query dari pengguna, sehingga diperoleh nilai pencarian dokumen yang paling efektif dan relevan dengan keinginan pengguna.

2.2 Efektifitas Pencarian Dokumen

Pengertian efektivitas adalah suatu ukuran yang menyatakan seberapa jauh target (kuantitas, kualitas, dan waktu) yang telah dicapai, yang mana target tersebut sudah ditentukan terlebih dahulu. Berdasarkan hal tersebut maka untuk mencari tingkat efektivitas dapat digunakan rumus sebagai berikut:

Efektivitas = (*Ouput* Aktual / *Output* Target = 1)

a. Jika *output* aktual berbanding *output* yang ditargetkan sama dengan 1 (satu), maka akan tercapai efektivitas
b. Jika *output* aktual berbanding *output* yang ditargetkan sama dengan 0 (nol), maka efektivitas tidak tercapai

Pengukuran efektivitas dalam *information retrieval* dapat dilakukan dengan perhitungan terhadap nilai perolehan (*recall*) dan nilai ketetapan (*precision*). Perhitungan ini juga akan memberikan hasil keefektifan setelah membandingkan dengan penelitian lain yang terkait dengan penelitian ini tetapi menggunakan metode yang berbeda.

2.3 Metode Analisa

Precision dapat dilihat sebagai ukuran ketepatan, sedangkan *recall* adalah ukuran kelengkapan. Dalam

sebuah pencarian informasi, *precision* didefinisikan sebagai jumlah dokumen yang relevan diambil oleh pencarian dibagi dengan total jumlah dokumen yang diambil oleh sistem pencari, sedangkan *recall* didefinisikan sebagai jumlah dokumen yang relevan diambil oleh pencarian dibagi dengan total jumlah dokumen yang relevan yang ada.

Dalam pencarian informasi, *precision* mempunyai nilai sempurna dengan nilai 1, ini berarti bahwa setiap hasil diambil dengan pencarian yang relevan (menyatakan semua dokumen yang relevan diambil) sedangkan nilai *recall* yang sempurna adalah 1 juga, ini berarti bahwa semua dokumen yang relevan diambil oleh search (menyatakan berapa banyak dokumen yang tidak relevan juga diambil).

Biasanya, nilai *precision* dan *recall* tidak dibahas secara terpisah. Sebaliknya, nilai-nilai baik untuk satu ukuran akan dibandingkan untuk tingkat yang tetap pada ukuran lain (misalnya *Precision* pada tingkat penarikan kembali 0,75) atau keduanya digabungkan menjadi satu ukuran.

Dalam *information retrieval, precision* dan *recall* ditetapkan dalam bentuk satu set dokumen yang diambil (misalnya daftar dokumen yang dihasilkan oleh sebuah web *search engine* untuk pencarian) dan satu set dokumen yang relevan (misalnya daftar semua dokumen didalam koleksi dokumen yang relevan untuk topik tertentu).

Dalam bidang pencarian informasi, precision adalah bagian dari dokumen yang diambil relevansinya dengan pencarian.

$$Precision = \frac{\textbf{jumlah dokumen relevan yang berhasil ditemukan}}{\textbf{jumlah seluruh dokumen yang ditemukan}}$$

Precision mengambil semua dokumen dan hitung, tetapi juga dapat dievaluasi pada suatu pembatasan peringkat, hanya mempertimbangkan hasil paling atas yang dikembalikan oleh sistem. Sebagai contoh untuk pencarian teks pada dokumen, *precision* adalah jumlah hasil benar dibagi dengan jumlah semua hasil kembali.

Recall dalam *Information Retrieval* adalah bagian dari dokumen-dokumen yang relevan dengan query yang berhasil diambil.

$$Recall = \frac{\textbf{jumlah dokumen relevan yang berhasil ditemukan}}{\textbf{jumlah seluruh dokumen relevan}}$$

Sebagai contoh untuk pencarian teks pada kumpulan dokumen *recall* adalah jumlah hasil yang benar dibagi dengan jumlah hasil yang seharusnya dikembalikan. Dalam klasifikasi biner, *recall* disebut kepekaan. Sehingga dapat dipandang sebagai probabilitas bahwa sebuah dokumen yang relevan diambil oleh query.

2.4 Variable Penelitian

Variabel-variabel yang digunakan pada penelitian ini adalah variabel yang berupa variable kuantitatif untuk untuk mengukur bobot dokumen (X_{jl}) dan bobot *query* (X_{jk}) serta korelasi antara bobot-bobot tersebut. Dalam hal ini variable yang digunakan adalah nilai keefektifan pencarian dokumen (*Precision* dan *recall*)

3. HASIL DAN DISKUSI

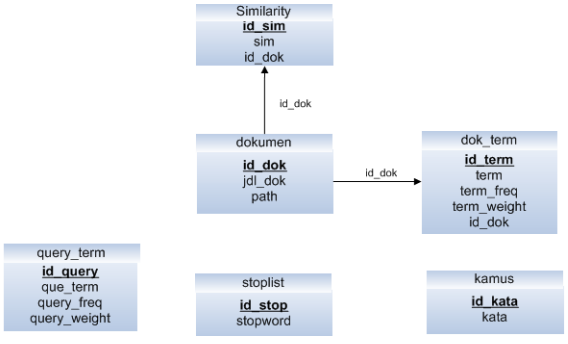
3.1 Arsitektur Proses

Alur proses pada sistem yang digunakan pada penelitian ini dibagi menjadi dua proses yang terpisah tetapi nantinya akan disatukan pada perhitungan bobot, yaitu proses penyimpanan dokumen dan query. Dari proses tersebut terdapat beberapa tahap perolehan hasil diantaranya yaitu input, *extract*, *tokenization*, *stopword, stemming*, database, *term weighting*, dan *similarity measurement*.

Pada proses penyimpanan dokumen, disiapkan dokumen yang akan dijadikan sebagai koleksi dokumen. Selanjutnya dokumen tersebut diekstrak untuk memisahkan antara teks dengan *tag* html serta untuk mendapatkan urutan yang lebih terstruktur pada dokumen tersebut. Lalu dokumen yang telah diekstrak masuk kedalam tahap kedua, yaitu *tokenization*. Dalam proses ini, dokumen teks yang telah diekstrak dibuat menjadi perbagian, yaitu pembuangan karakter tertentu seperti tanda baca. Setelah pembuangan karakter, tahap selanjutnya adalah proses *stopword removal*. Pada proses ini, kata-kata yang tidak perlu dibuang dengan cara membandingkannya dengan stoplist yang ada. Lalu tahap selanjutnya adalah stemming, pada tahap ini kata-kata yang telah melalui tahap stopword removal dijadikan kata dasar yang kemudian nantinya disimpan kedalam database sebagai kumpulan *term*.

Pada proses yang kedua, proses yang berjalan hampir sama dengan proses yang terjadi pada proses pertama, hanya bedanya terdapat pada tidak adanya proses *extract*, untuk proses dari input sampai dengan masuk kedalam database. Setelah hasil query tersebut masuk kedalam database, proses selanjutnya adalah penghitungan bobot dari masing-masing *term* pada dokumen dan *term* pada query. Setelah mendapatkan bobot dari masing-masing *term* tersebut, selanjutnya adalah proses perhitungan kesamaan (*similarity measurement*) antara bobot dokumen dengan bobot query. Setelah mendapatkan hasil *similarity*, selanjutnya adalah proses pemeriksaan nilai efektivitas.

3.2 Arsitektur Basis Data



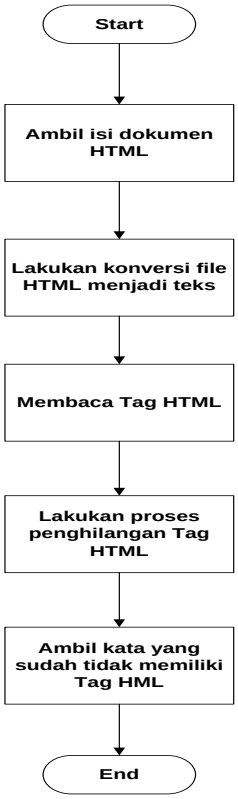
Gambar 2 : Logical Record Structure

Pada database yang dibuat terdapat 6 tabel yang 3 diantaranya memiliki relasi satu dengan yang lainnya. Yaitu tabel dokumen yang berisi daftar koleksi dokumen, tabel dok_term (dokumen *term*) berupa tabel yang berisi daftar *term* yang diperoleh dari dokumen serta tabel *similiarity* yang terdiri nilai *similarity* dokumen dengan query. Tabel dokumen sendiri memiliki relasi dengan tabel dok_term dan tabel *similiarity*. Sedangkan 3 tabel lain yaitu tabel query_term, *stoplist*, kamus tidak memliki relasi dengan tabel lainnya. Tabel query *term* merupakan tempat penyimpanan query. Kemudian tabel stoplist dan kamus berfungsi sebagai tempat penyimpanan daftar kata stopword & kata dasar.

Sehingga jika dikaitkan dengan alur program maka setiap kali dokumen baru dimasukan, tabel dokumen akan bertambah sebanyak daftar koleksi dokumen baru kemudian setelah dokumen tersebut melalui proses *parsing* & *stemming* dimana sistem akan menggunakan tabel stoplist & kamus. Setelah itu hasil proses akan dimasukan kedalam tabel dok_term dengan id dokumen baru. Ketika query dimasukan,query tersebut akan disimpan sementara pada tabel query_term. Sedangkan proses terakhir akan melibatkan tabel dokumen dengan tabel query_term dalam perhitungan nilai similarity dokumen dengan query yang akan disimpan pada tabel similiarity.

3.3 Parsing

Tahap pertama yang harus dilakukan adalah mengambil dokumen dengan format HTML yang disimpan dalam sebuah alamat atau lokasi yang sudah ditentukan sebelumnya. Selanjutnya dokumen yang diperoleh akan mengalami proses parsing. Tahap awal yang pada proses parsing adalah memasukan alamat dan nama dokumen kedalam database, kemudian pada tahap selanjutnya sistem akan mengambil content dari tiap dokumen yang sesuai dengan alamat dan nama dokumen yang akan mengalami proses parsing. Langkah proses parsing adalah konversi file dari yang berformat HTML menjadi teks. Kemudian program parsing tersebut akan membaca tag HTML yang ada pada dokumen tersebut Setelah proses membaca *tag* HTML, tahap berikutnya adalah proses menghilangkan tag tersebut dengan cara mencari tanda tag “<>” dan “</>” pada setiap baris teks. Apabila dalam proses pembacaan baris teks tersebut ditemukan tanda *tag*, maka tag tersebut dihapus. Proses ini memiliki tujuan untuk memisahkan *content* dokumen dengan *tag* HTML agar informasi yang terdapat dalam dokumen lebih terstruktur. Berikut alur program Parsing HTML



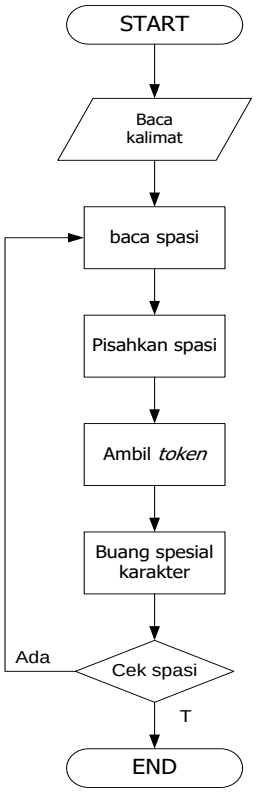
Gambar 3 :Flowchart Proses Parsing

Berikut adalah penjelasan dari proses parsing diatas :

a. Mengambil dokumen HTML dari koleksi dokumen yang telah disiapkan sebelumnya.
b. membuat dan memanggil fungsi untuk mengkonversi HTML menjadi teks.
c. Melakukan proses membaca tag HTML dengan fungsi HTML Parser.
d. Setelah mengenali semua tag HTML maka dilakukan proses penghilangan tag HTML.
e. Proses terakhir adalah menampung kata yang sudah tidak memiliki tag HTML sebagai index\

3.4 Tokenization

Pada tahap selanjutnya akan dilakukan proses tokenization, pada proses ini dokumen akan dibuat menjadi dalam bentuk string untuk satu kata tunggal. Selain itu *tokenization* juga digunakan untuk menghilangkan tanda baca dan karakter spesial sehingga nantinya akan memudahkan proses pemotongan kata berimbuhan kedalam bentuk kata dasar pada proses stemming. Flowchart proses *tokenization* dapat dilihat pada gambar berikut



Gambar 4 : Flowchart Proses Tokenization

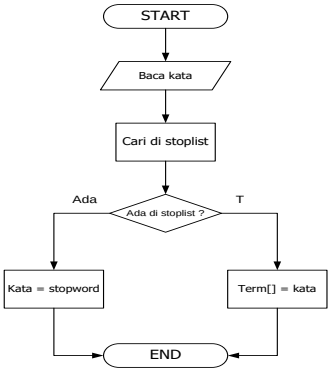
Berikut adalah penjelasan dari proses tokenization diatas :

a. Mengambil kalimat hasil parsing dokumen pada proses sebelumnya.
b. Membaca spasi yang ada pada kalimat.
c. Memisahkan spasi agar tersisa token tunggal.
d. Setelah token berhasil dipisahkan dari spasi maka dilakukan penghilangan karakter spesial.
e. Kemudian diperiksa kembali apakah ada spasi pada kalimat berikutnya. Kalau ada tahap kedua diulang kembali dan seterusnya

3.5 Stop Word Removal

Proses selanjutnya proses *stopword removal*, yaitu proses pembuangan kata umum yang tidak memiliki makna seperti kata “yang”, “ke”, “di”, “dengan”, “seperti”, “tidak”, “dan”, dsb. Pada proses ini, kata – kata yang telah melewati proses *tokenization* akan dibandingkan dengan daftar kata yang berada pada tabel *stoplist* dalam database. Jika kata tersebut ada dalam tabel *stoplist*, maka kata tersebut akan dianggap dan ditandai sebagai *stopword*.

Berikut adalah *flowchart* dari proses *stopword removal*:



Gambar 5 : Flowchart Proses Stopword Removal

Kata – kata yang telah ditandai sebagai *stopword* tidak akan diikutsertakan dalam proses selanjutnya, yaitu *stemming*, sehingga dapat meringankan kerja sistem. Disamping itu, ukuran indeks *term* pada *database* tidak akan terlalu besar. Apabila kata – kata yang ditandai sebagai *stopword* tetap diikutsertakan dalam proses *stemming*, maka kerja sistem akan semakin berat dan membutuhkan *resource* yang besar pula. Hal ini dikarenakan oleh kata – kata yang sebenarnya tidak diperlukan ikut menambah ukuran indeks *term.*

3.6 Steming

Stemming merupakan sebuah teknik yang bertujuan untuk mereduksi bentuk dari sebuah kata berimbuhan menjadi bentuk gramatikalnya atau dasarnya [2]. Sebagai contoh untuk *Stemming* dalam bahasa Indonesia pemotongan imbuhan dapat dilakukan pada awalan (*prefixes*), akhiran (*suffixes*), ataupun sisipan (*infixes*), seperti kata "berbelanja" memiliki bentuk dasar "belanja", kata "menjalankan" memiliki bentuk dasar "jalan", dan yang lainnya. Dengan melakukan stemming, ukuran indeks dapat diperkecil karena beberapa kata berimbuhan yang memiliki kata dasar yang sama akan diindeks menjadi satu.

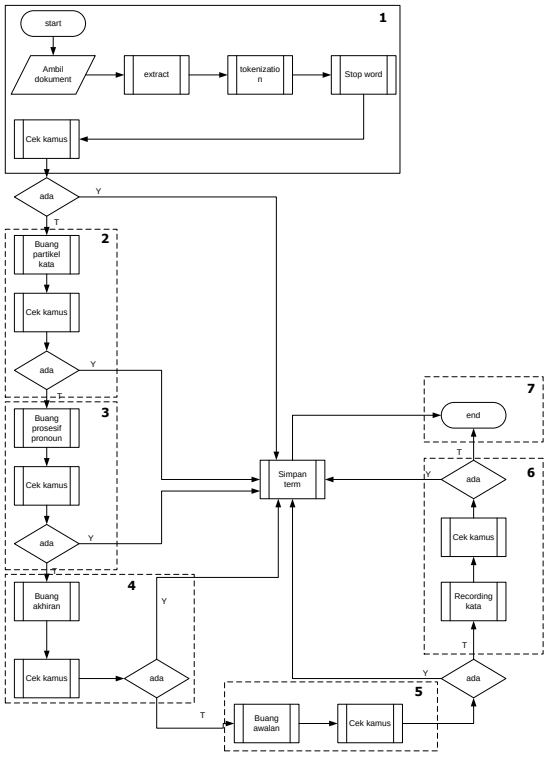
Stemming dilakukan atas dasar asumsi bahwa kata-kata yang memiliki stem yang sama memiliki makna yang serupa sehingga pengguna tidak keberatan untuk memperoleh dokumen-dokumen yang didalamnya terdapat kata-kata dengan *stem* yang sama dengan querynya.

Proses ini memiliki dua tujuan. Dalam hal efisiensi, *stemming* mengurangi jumlah kata unik dalam indeks sehingga mengurangi kebutuhan ruang penyimpanan untuk indeks dan mempercepat proses pencarian. Hal tersebut tidak akan diperoleh jika tiap bentuk suatu kata disimpan secara terpisah dalam indeks. Akan tetapi, *stemming* dapat menurunkan tingkat precision jika setiap bentuk suatu *stem* diperoleh, sedangkan yang relevan hanyalah bentuk yang sama dengan yang digunakan dalam query [1].

Prosedur *stemming* yang diterapkan pada penelitian ini adalah perpaduan Algoritma Porter dan Algoritma Nazief and Adriani [6]. Perpaduan dilakukan untuk mengsiasati kesulitan dalam proses

ini dikarenakan struktur bahasa Indonesia yang berbeda dengan bahasa Inggris (jika hanya menggunakan algoritma Porter). Selain itu juga dikarenakan dokumen yang digunakan dalam penelitian ini semuanya berbahasa Indonesia.

Untuk lebih jelasnya, proses *stemming* yang digunakan dalam aplikasi ini akan dijelaskan dalam *flowchart* berikut ini



Gambar 6 : Flowchart Proses Stemming

Gambar diatas merupakan gambar alur proses *stemming*, baik *steming* query maupun dokumen pada aplikasi yang dibuat dalam penelitian ini, dengan penjelasan sebagai berikut :

a. Ketika sebuah dokumen diupload, maka dokumen akan langsung diekstrak untuk mengambil kata - kata isi dokumen. Setelah itu akan dilakukan proses *tokenization* yaitu penghilangan tanda baca, setelah kata - kata murni didapat maka akan dilakukan penghilangan *stopword* lalu akan diperiksa kedalam kamus. Kemudian kata - kata yang belum di-*stemming* tersebut dicari dalam kamus. Jika kata - kata dalam dokumen itu langsung ditemukan, berarti kata - kata tersebut adalah kata dasar dan akan langsung dimasukkan kedalam tabel *term* pada database.
b. Dan apabila tidak ada dalam kamus, maka akan dilakukan proses penghilangan partikel kata (lah, tah, kah, pun) lalu akan diperiksa kembali kedalam kamus. Apabila kata terdapat dalam kamus maka akan langsung disimpan sebagai *term*.
c. Apabila bila kata masih juga belum sesuai dengan kamus, maka akan dilakukan penghilangan posesif pronoun (ku, mu, dan nya) lalu akan diperiksa kembali kedalam kamus, apabila kata tersebut sesuai dengan kamus maka akan langsung disimpan sebagai *term*.
d. Jika tidak sistem akan melakukan penghilangan akhiran kata, setelah itu akan dilakukan diperiksa kembali kedalam kamus. Jika kata yang telah dihilangkan akhirannya sesuai dengan kamus maka kata itu akan langsung dimasukan sebagai *term*.
e. Apabila didalam pengecekan kedalam kamus diatas masih tidak sesuai dengan kamus, maka akan dilakukan penghilangan awalan dan dilakukan pengecekan kembali kedalam kamus. Jika kata tersebut sesuai dengan kamus maka akan langsung disimpan sebagai *term*.
f. Setelah tidak ada lagi imbuhan yang tersisa, maka algoritma ini dihentikan kemudian kata dasar tersebut dicari pada kamus, jika kata dasar tersebut ketemu berarti algoritma ini berhasil tapi jika kata dasar tersebut tidak ketemu pada kamus, maka dilakukan *Recoding* (pengembalian hasil dari pemotongan, stemming). Setelah itu akan dilakukan pengecekan kembali kedalam kamus, jika ada maka akan langsung disimpan sebagai *term*.
g. Jika semua langkah telah dilakukan tetapi kata dasar tersebut tidak ditemukan pada kamus juga maka kata tersebut digolongkan sebagai *term* dengan jumlah nol kedalam database

3.7 Representasi Vektor

Vektor merupakan salah satu model dari *information retrieval* yang mampu menghasilkan dokumen-dokumen terurut berdasarkan kesesuaian dengan query yang dicari. Misalkan terdapat sejumlah n kata yang berbeda sebagai kamus kata (vocabulary) atau index kata (terms index). Kata-kata ini akan membentuk ruang vektor yang memiliki dimensi sebesar n. Setiap kata i dalam dokumen atau query diberikan bobot sebesar Win. Baik dokumen maupun query direpresentasikan sebagai vektor berdimensi n.

Model ruang vektor untuk koleksi dokumen mengandaikan dokumen sebagai sebuah vektor dalam ruang kata (feature). Jika koleksi n buah dokumen dapat diindeks oleh fin buah term/feature maka suatu dokumen dapat dipandang sebagai vektor berdimensi fin dalam ruang term tersebut. fin merupakan frekuensi term i didalam satu dokumen. Koleksi dokumen diwakili oleh beberapa dokumen yang tersimpan didalam database. Sebagai simulasinya sebagai berikut :

a. Terdapat tiga buah dokumen (D1, D2, D3) yang telah melalui proses extract, tokenization, stopword removal, dan stemming seperti proses yang telah dijelaskan pada bagian sebelunya menghasilkan indeks term yang terdapat pada database.
b. Index term pada database yang telah terbentuk dapat dilihat pada tabel matriks berikut :

Term space	Frekuensi *Term*		
	D1	**D2**	**D3**
budi	3	2	0
main	2	0	4
bola	2	2	1

Tabel 1 : Ilustrasi Tabel Simulasi Vektor

3.8 Prosedur Pengukuran

Terdapat dua jenis prosedur pengukuran, yaitu pengukuran pembobotan suatu dokumen terhadap sebuah *term* dan pengukuran *similarity measurement* (kesamaan antara bobot suatu dokumen terhadap query).

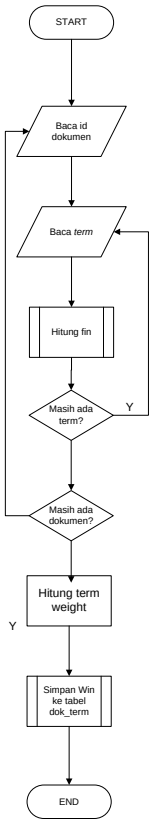
Pada pengukuran bobot atau *term weighting*, ada dua macam, yaitu pembobotan query dan pembobotan suatu dokumen terhadap suatu term. Pada pembobotan query, menggunakan model *binary term*. Model ini dimaksudkan untuk *term* query mempunyai nilai 1. Sedangkan untuk menghitung bobot term didalam dokumen dengan menggunakan formula sebagai berikut

$$W_{in} = \log(f_{in})$$

Keterangan :

- W_{in} : nilai bobot suatu term terhadap sebuah dokumen
- f_{in} : frekuensi *term* i didalam dokumen n

Untuk lebih jelasnya, berikut alur program pembobotan term pada dokumen



Gambar 7 : Flowchar Proses Pembobotan

Program perhitungan bobot dokumen *term* ada pada *class Weight* (file = weight.php) fungsi hit_term_weight().

a. f_{in} -> sudah ada pada table dok_term karena proses install. (extract, stop + stemming, simpan *term*, hitung term_freq).
b. Mengambil *field* term, id_dok, term_freq dari table dok_term menggunakan *sql query*.
c. Hasil query tadi disimpan kedalam *array* 2 dimensi.
d. Lakukan perulangan / loop untuk masing-masing isi elemen *array*.
e. Isi elemen term_freq pada *array* tersebut disimpan pada variable $fin.
f. Ambil jumlah isi dari *field* term_freq pada table dok_term berdasarkan id_dok untuk menghitung fi menggunakan fungsi SUM pada *sql query*.
g. Jumlah isi field term_freq dimasukkan kedalam variable yang bernama $fi.
h. Lakukan perhitungan weight dengan rumus $tw = round($fin/$fi,4). Maksudnya round(x,n) adalah fungsi php utk membulatkan bilangan desimal (x) sebanyak n angka diblakang koma. Karena memakai pembulatan 4 angka dibelakang koma maka dipakai n=4.
i. Ubah field term_weight pada table dok_term dengan angka hasil perhitungan tadi dengan menggunakan *sql query*.
j. Langkah 5-9 diulang sebanyak jumlah *term* pada satu dokumen dan diulang kembali sebanyak jumlah dokumen didalam koleksi dokumen.
k. Isi field term_weight pd table dok_term akan berubah sesuai hasil perhitungan.

Sokal Similarity Measurement :

$$Sim\ (Q,D) = \frac{\sum(X_{jk}\,.\,X_{jl})}{2\sum(X_{jk})^2 + 2\sum(X_{jl})^2 - 3\sum(X_{jk}\,.\,X_{jl})}$$

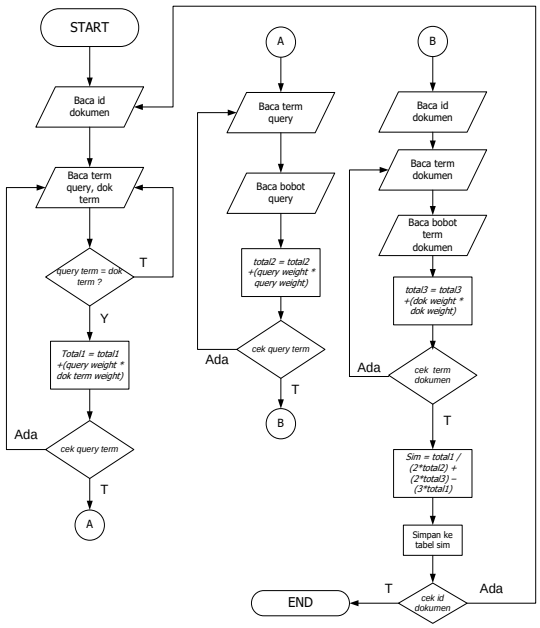
Keterangan :

- Sim (Q,D) : *similarity* antara *query* dengan dokumen
- X_{jk} : bobot term *query*
- X_{jl} : bobot term dokumen

Setelah mendapatkan nilai *term weight*, tahap selanjutnya adalah menghitung *similarity*. Untuk mempermudah penuangan rumus kedalam program maka pada program dibagi per bagian perhitungan. Karena rumus ini berlaku untuk setiap dokumen maka proses dilakukan berulang kali sebanyak jumlah dokumen yang ada pada koleksi. Caranya dengan mengambil isi *field* id_dok dari table dokumen dengan menggunakan *sql query*.

Pada perhitungan *similarity*, proses dibagi menjadi empat tahapan. Tahap yang dilakukan adalah menghitung $\sum(X_{jk}.X_{jl})$, kemudian menghitung $\sum(X_{jk})^2$, dan yang berikutnya menghitung $\sum(X_{jl})^2$. Terakhir adalah menggabungkan hasil-hasil

perhitungan kedalam bentuk rumus *Sokal*. Berikut ini adalah *flowchart* dari proses – proses tersebut.



Gambar 8: Flowchar Proses Hitung Similarity

Perhitungan hasil *similarity* dalam program dapat diuraikan sbb :

a. Ambil isi dari field term, term_weight dari tabel dok_term dan query_term, query_weight dari table query_term berdasarkan ID dokumen yang sedang dilakukan perhitungan.
b. Isi field-field yang sudah diambil dimasukkan kedalam array.
c. Karena tiap-tiap dokumen kemungkinan memiliki term lebih dari 1 dan bagian perhitungan ini merupakan penjumlahan ($\sum$), maka pada program dilakukan perulangan untuk menghitung hasil perkalian antara bobot term *query* dan bobot dokumen term ($X_{jk} . X_{jl}$).
d. Satu variable disiapkan untuk menampung hasil perkalian tadi dan selama perulangan isi variable bertambah terus sehingga hasil perkaliannya akan berbentuk sigma.
e. Tahap perhitungan $\sum(X_{jk})^2$ adalah proses perhitungan jumlah hasil pembobotan kuadrat term-term *query*.
f. Isi field query_weight dari table query_term diambil menggunakan *sql query* dan dimasukkan kedalam variabel *array*.
g. Tiap-tiap elemen *array* yang isinya bobot term *query* dihitung kuadratnya kemudian dijumlahkan.
h. Hasilnya disimpan kedalam satu variabel baru.
i. Tahap perhitungan $\sum(X_{jl})^2$ sama dengan tahap perhitungan $\sum(X_{jk})^2$ tetapi yang menjadi perbedaan hanya field yang diambil adalah term_weight dari table dok_term.
j. Hasil – hasil perhitungan kemudian diformulasikan kedalam bentuk rumus *Sokal*.
k. Hasil perhitungan rumus *Sokal* disimpan kedalam variabel dan dimasukkan kedalam tabel *similarity*.

Isi dari variable sim kemudian disimpan kedalam table *similarity* masing-masing per dokumen. Setelah semua hasil similarity masuk kedalam table maka akan terlihat angka-angka *similarity query* terhadap masing-masing dokumen

Dari hasil perhitungan *similarity* tersebut diperoleh nilai *similarity* dari masing-masing dokumen terhadap query yang dimasukkan. Berdasarkan nilai – nilai *similarity* yang didapat

Setelah mendapatkan hasil *similarity*, tahap selanjutnya adalah menghitung *precision* dan *recall* dengan cara manual untuk mendapatkan nilai efektivitas dari model pembobotan dan *similarity* yang digunakan

3.9 Model Evaluasi

Pada tahap ini yang akan dibahas adalah model evaluasi terhadap sistem yang telah dibangun untuk mencapai kesimpulan akhir dari penelitian yang dilakukan. Model evaluasi yang digunakan adalah model pencarian nilai efektivitas dokumen terhadap query dengan menggunakan *precision* dan *recall*.

Evaluasi dan analisa pada sistem dilakukan dengan beberapa tahap. Setelah proses penyiapan koleksi dokumen, maka kumpulan dokumen tersebut diekstrak dan kumpulan *term* yang dihasilkan disimpan pada database. Indeks *term* akan terbentuk setelah kumpulan term dokumen berhasil disimpan dalam database.

Setelah indeks *term* berhasil terbentuk, proses input query dapat dilakukan. Query yang dimasukkan telah ditetapkan sebanyak 10 kali input berdasarkan topik yang ditentukan. Tiap kali query diinput, maka sistem akan menampilkan dokumen-dokumen beserta nilai *similarity* dokumen tersebut terhadap query yang dimasukkan.

Pada tahap ini, proses perhitungan *precision* dan *recall* dapat mulai dilakukan. Caranya adalah dengan melihat dokumen-dokumen yang relevan berdasarkan topik lalu dihitung nilai *precision* dan *recall* sistem pada penelitian ini. Setelah perhitungan *precision* dan *recall* pada masing – masing query yang dimasukkan, langkah evaluasi selanjutnya adalah menghitung rata-rata *precision* dan *recall* dari hasil perhitungan sebelumnya. Hasil perhitungan rata-rata *precision* dan *recall* kemudian dituangkan dalam bentuk grafik.

Selain itu, terdapat pula model evaluasi perbandingan hasil efektivitas yang diperoleh dengan nilai efektivitas penelitian yang lainnya. Hasil evaluasi yang digunakan adalah nilai rata-rata dari setiap *precision per recall* dan digambarkan dalam bentuk grafik.

3.10Evaluasi Sistem

Koleksi dokumen terdapat 100 dokumen yang terdiri atas 10 macam topik yang berbeda-beda yang terkandung pada informasi didalamnya. Masing-

masing topik terdiri dari 10 dokumen / artikel. Topik-topik yang akan digunakan telah ditentukan seperti pada tabel berikut.

No	Topik	Jumlah	Format	*Query* yang diinput
1.	film 2012	10	HTML	"pembuatan film 2012"
2.	banjir	10	HTML	"bencana banjir"
3.	haji	10	HTML	"pemeriksaan haji"
4.	islam	10	HTML	"penyebaran islam"
5.	pariwisata	10	HTML	"kota pariwisata"
6.	musik	10	HTML	"penonton konser musik"
7.	bunuh diri	10	HTML	"korban bunuh diri"
8.	kebakaran	10	HTML	"penyebab kebakaran"
9.	aksi antikorupsi	10	HTML	"cegah korupsi"
10.	HIV/AIDS	10	HTML	"penyebaran hiv"

Tabel 2 : Tabel Topik pada Koleksi Dokumen

Setelah nilai *precision* dan *recall* pada masing – masing query yang dimasukkan dan rata-rata *precision* dan *recall* telah dihitung, maka berikut ini adalah tabel hasil perhitungan yang dilakukan pada tahap evaluasi ini

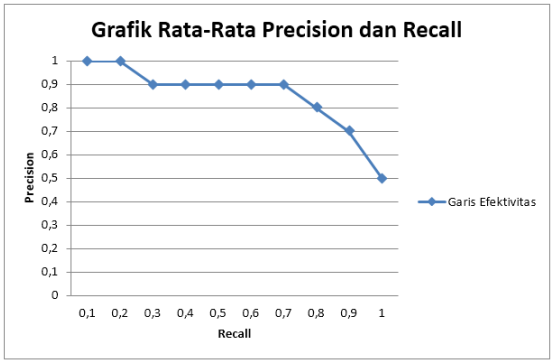
No	Rata - Rata	
	Precc	Recall
1	1	0,1
2	0,97	0,2
3	0,95	0,3
4	0,94	0,4
5	0,91	0,5
6	0,90	0,6
7	0,89	0,7
8	0,74	0,8
9	0,69	0,9
10	0,55	1

Tabel 3 : Tabel Rata-Rata Precision dan Recall

Berdasarkan tabel diatas dapat dilihat bahwa pada setiap nilai *recall*, nilai *precision* bersifat variatif (tidak selalu konstan). Hal ini disebabkan oleh beberapa faktor berikut ini

a. Pada setiap input query, jumlah dokumen yang berhasil ditemukan oleh sistem bervariasi.
b. Ada beberapa dokumen yang memiliki nilai *similarity* terhadap query yang cukup tinggi tetapi sebenarnya dokumen tidak relevan dengan query.
c. Tidak semua input query menghasilkan urutan dokumen dimana 10 dokumen yang topiknya relevan dengan query. Ada dokumen yang tidak relevan masuk pada urutan 10 teratas sehingga dokumen yang relevan berada pada urutan dibelakang.
d. Ada dokumen yang relevan dengan query tetapi tidak terambil oleh sistem

Faktor-faktor yang telah disebutkan diatas akan menurunkan nilai *precision* pada setiap nilai *recall*. Berikut ini adalah grafik rata-rata *precision* dan *recall* hasil pencarian dari query yang dimasukkan berdasarkan tabel diatas



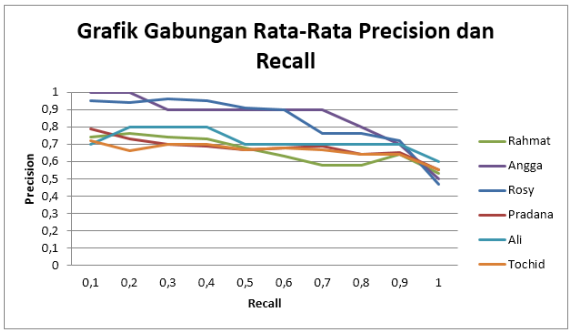
Gambar 9 : Grafik Rata-Rata Precision dan Recall

3.11Perbandingan Dengan Penelitian Terkait

Pada penelitian ini, ada beberapa penelitian lain yang terkait dengan penelitian yang dilakukan, perbedaannya terletak pada pembobotan dan *similarity* yang digunakan. Penelitian-penelitian tersebut antara lain penerapan metode pembobotan berdasarkan pembagian antara jumlah satu *term* didalam satu dokumen dengan jumlah frekuensi *term* yang muncul didalam suatu dokumen dikalikan dengan jumlah *term* tersebut dalam database dan metode pengukuran kesamaan *Sokal Coefficient* pada penelitian Rahmat Oktavian [7], penerapan metode pembobotan *based on term frequency* dengan pengukuran *Jaccard* oleh Rosy Marselina [5], evaluasi kuantitatif hasil pencarian dokumen HTML dengan metode pembobotan berdasarkan pembagian antara jumlah satu *term* dalam satu dokumen dengan jumlah *term* tersebut dalam database dan metode pengukuran kesamaan *Sokal Coefficient* oleh Ali Suwanda [10], penerapan metode pembobotan berdasarkan pembagian antara jumlah satu *term* didalam satu dokumen dengan jumlah *term* tersebut didalam database dan pengukuran *Sokal* oleh Tochid [12], dan penerapan metode pembobotan berdasarkan pembagian antara jumlah satu *term* didalam satu dokumen dengan jumlah *term* tersebut didalam database dan metode pengukuran kesamaan *Dice Coefficient* oleh Rizky Pradana [8].

Langkah-langkah penelitian yang dilakukan pada penelitian-penelitian terkait diatas sama dengan penelitian ini. Data yang digunakan dalam eksperimen juga sama dengan yang digunakan pada penelitian ini (100 dokumen dengan 10 topik yang berbeda dan 10 masukan query yang sama). Hal ini dilakukan agar tidak terjadi perbedaan dalam lingkup penelitian antara penelitian yang dilakukan ini dengan penelitian lain yang terkait.

Berikut hasil evaluasi penelitian yang telah disatukan dengan peneliti lain dalam bentuk grafik rata-rata *precision* dan *recall*



Gambar 10 : Grafik Gabungan Rata-rata Precision dan Recall

Percobaan yang dilakukan dengan 10 query dan 100 dokumen pada penelitian ini menghasilkan nilai rata- rata *precision* dengan *range* 0,85 – 1 seperti pada tabel sebelumnya. Bila dibandingkan dengan hasil penelitian lain yang terkait, bisa dilihat bahwa penelitian yang menggunakan metode pembobotan pembagian antara jumlah frekuensi suatu *term* didalam satu dokumen dengan jumlah seluruh frekuensi *term* dalam dokumen tersebut dan dengan metode *Sokal similarity measurement* menempati garis teratas pada grafik gabungan rata – rata *precision* dan *recall* tapi kemudian cenderung menurun.

4. KESIMPULAN

Dari penggunaan pembobotan logaritma frekuensi satu *term* pada dokumen dan dengan pengukuran kesamaan menggunakan metode sokal dapat disimpulkan bahwa semakin besar nilai *recall*, maka nilai presisinya cenderung semakin kecil. Selain itu apabila dibandingkan dengan beberapa kombinasi metode pembobotan dan pengukuran kesamaan yang lain, penggunaan metode pembobotan pembagian antara jumlah frekuensi satu *term* didalam satu dokumen dengan jumlah frekuensi *term* tersebut didalam database dan pengukuran kesamaan *sokal* berada pada tingkat keefektifan tinggi.

Penerapan metode penelitian ini, baik diterapkan pada sistem pengarsipan dokumen, karena akan mendapatkan hasil pencarian yang berurut, dari dokumen yang paling banyak mengandung term query hinggan dokumen yang paling sedikit mengandung term query.

Dalam penelitian lanjutan, diharapkan penelitian dapat memfokuskan pada hal-hal yang belum diteliti seperti bagaimana mentukan relevansi antara query dengan topik secara otomatis sehingga pengguna tidak perlu menentukan apakan judul yang muncul tersebut relevan atau tidak dengan yang dikehendakinya. Penelitian lanjutan ini merupakan suatu tantangan yang sangat sulit, karena dalam menentukan relevan atau tidaknya selama ini hanya dengan mencocokkan secara manual judul yang muncul sesaat setelah pencarian dokumen dengan topik yang telah terlebih dahulu telah mengelompokkan artikel-artikel kedalam beberapa topik. Sehingga dapat mempercepat perhitungan *precesion* dan *recall* jika memiliki koleksi dokumen yang cukup banyak.

5. PUSTAKA

[1] Ellis D., et. al., 1994, *Measuring the Degree of Similarity Between Objects in Text-retrieval Systems*. Perspective of Information Management, 3:128-149.

[2] Frakes, W., and R. Baeza-Yates, Eds. 1992, New Jersey, Prentice-Hall, *Information Retrieval: Data Structures and Algorithms.*

[3] Korfhage, R., 1997, New York, *Information Storage and Retrieval*.

[4] Lee, Yuan., 1992, Miami, Miami University, *Design and Implementation of a Hypertext-based Information Retrieval System*.

[5] Marselina, Rosy, 2010, Jakarta, *Evaluasi Kuantitatif Efektifitas Hasil Pencarian Dokumen Dengan Menggunakan Jaccard Coeficient : Suatu Studi Kasus Penerapan Stemmed Term Vektor Model Untuk Representasi Dokumen Yang Menggunakan Bobot Jumlah Satu Term Didalam Satu Dokumen pada Mesin Pencari Berbasis HTML*.

[6] Nazief, B., and M. Adriani, 1996, Jakarta, Universitas Indonesia, *Confix Stripping : Approach to Stemming Algorithm for Bahasa Indonesia*.

[7] Oktavian, Rahmat, 2010, Jakarta, *Evaluasi Kuantitatif Efektifitas Hasil Pencarian Dokumen Dengan Menggunakan Cosine Coeficient : Suatu Studi Kasus Penerapan Stemmed Term Vektor Model Untuk Representasi Dokumen Yang Menggunakan Bobot Pembagian Antara Jumlah Satu Term Didalam Satu Dokumen Dengan Jumlah Frekuensi Term yang Muncul Didalam Suatu Dokumen Dikalikan Dengan Jumlah Term Tersebut Didalam Database pada Mesin Pencari Berbasis HTML*.

[8] Pradana, Rizky, 2010, Jakarta, *Evaluasi Kuantitatif Efektifitas Hasil Pencarian Dokumen Dengan Menggunakan Dice Coeficient : Suatu Studi Kasus Penerapan Stemmed Term Vektor Model Untuk Representasi Dokumen Yang Menggunakan Pembobotan Pembagian Antara Jumlah Satu Term Didalam Satu Dokumen Dengan Jumlah Term Tersebut Didalam Database pada Mesin Pencari Berbasis HTML*.

[9] Salton, Gerard., and Chris Buckley, 1987, New York, Cornell University, *Term Weighting Approaches in Automatic Text Retrieval System*.

[10] Suwanda, Ali, 2010, Jakarta, *Evaluasi Kuantitatif Efektifitas Hasil Pencarian Dokumen Dengan Menggunakan Cosine Coeficient : Suatu Studi Kasus Penerapan Stemmed Term Vektor Model Untuk Representasi Dokumen Yang Menggunakan Bobot Pembagian Antara Jumlah Satu Term Didalam Satu Dokumen Dengan Jumlah Term Tersebut Didalam Database pada Mesin Pencari Berbasis HTML*.

[11] Tala, Fadilah Z., 2001, Amsterdam, Universiteit van Amsterdam, *A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia*.

[12] Tochid, 2010, Jakarta, *Evaluasi Kuantitatif Efektifitas Hasil Pencarian Dokumen Dengan Menggunakan Sokal Coeficient : Suatu Studi Kasus Penerapan Stemmed Term Vektor Model Untuk Representasi Dokumen Yang Menggunakan Bobot Pembagian Antara Jumlah Satu Term Didalam Satu Dokumen Dengan Jumlah Term Tersebut Didalam Database pada Mesin Pencari Berbasis HTML*.