Peningkatan Kinerja Pencarian Dokumen Tugas Akhir menggunakan Porter Stemmer Bahasa Indonesia dan Fungsi Peringkat Okapi BM25

Monica Widiasri
Program Studi Teknik Informatika
Universitas Surabaya
monica@staff.ubaya.ac.id

Ellysa Tjandra
Program Studi Teknik Informatika
Universitas Surabaya
ellysa@staff.ubaya.ac.id

Lisa Maria Chandra
Program Studi Teknik Informatika
Universitas Surabaya
S6114052@student.ubaya.ac.id

Abstrak - Proses pencarian dokumen yang menggunakan *information retrieval* akan menerima *query* dan mengembalikan dokumen yang relevan dengan *query* pencarian tersebut. Relevansi diperhitungkan dari relevansi kata pada *query* dan kumpulan dokumen yang dicari. Pada sistem pencarian yang tidak mempertimbangkan variasi morfologi kata mengakibatkan dokumen yang mempunyai kata yang merupakan variasi dari kata pada *query* tidak dianggap sebagai dokumen hasil pencarian. Proses *stemming* dilakukan untuk mengenali variasi morfologi tersebut, dengan cara melakukan perubahan pada kata-kata berimbuhan dengan cara penghapusan awalan dan akhiran suatu kata menjadi kata dasarnya. Proses *stemming* dilakukan pada proses *indexing*, sehingga akan mengurangi ukuran dari *index* file. Hal itu dapat mengurangi waktu pencarian dan kebutuhan memori. Dokumen hasil pencarian akan ditampilkan sesuai nilai peringkat relevansi dokumen dengan *query* yang diberikan. Pemberian peringkat dilakukan dengan cara memberikan bobot pada dokumen. Dokumen yang mempunyai relevansi kata yang tinggi dengan *query*, akan diberikan bobot yang lebih besar. Pada sistem pencarian Tugas Akhir pada Universitas X, belum dilakukan proses *stemming* dan *indexing*. Untuk meningkatkan kinerja pencarian Tugas Akhir tersebut akan ditambahkan proses *stemming* dan *indexing*, serta pengurutan peringkat dokumen hasil pencarian. Proses stemming menggunakan *porter stemmer* bahasa Indonesia karena dokumen TA yang dicari berbahasa Indonesia, proses *indexing* menggunakan *inverted index*. Serta pengurutan dokumen hasil menggunakan fungsi peringkat Okapi BM25. Dari hasil uji coba, proses *stemming* dan fungsi peringkat yang dilakukan memberikan hasil pencarian yang lebih baik sesuai relevansi *query*. Penggunaan *stemming* dan *inverted index* menghemat penggunaan memori serta dapat mempercepat proses pencarian secara signifikan.

Kata Kunci : *porter stemmer*, *inverted index*, okapi BM25.

I. PENDAHULUAN

Pencarian dokumen dengan menggunakan *query* memiliki permasalahan yaitu hasil pencarian hanya dapat ditemukan pada dokumen yang mempunyai kata yang sama dengan *query*. Dokumen yang mempunyai kata yang merupakan variasi dari kata pada *query* tidak dianggap sebagai dokumen hasil pencarian. Dalam dokumen yang ditulis dalam bahasa alami, sulit untuk mengambil informasi yang relevan karena bahasa memiliki berbagai varian morfologi kata-kata yang mengakibatkan terjadinya ketidaksesuaian kosakata [1]. Padahal, kata kunci yang mempunyai banyak variasi morfologi pada sistem pencarian informasi harus dipertimbangkan sebagai *term* yang sama [2]. Untuk mengenali variasi morfologi tersebut, maka dapat dilakukan proses *stemming*, yaitu melakukan perubahan pada kata-kata berimbuhan dengan cara penghapusan awalan dan akhiran suatu kata menjadi kata dasarnya. Kata yang memiliki kata dasar yang sama akan memiliki makna yang sama, sehingga stemming digunakan pada *information retrieval* untuk meningkatkan keakuratan perolehan informasi. Selain untuk meningkatkan keakuratan perolehan informasi, *stemming* yang dilakukan pada proses *indexing* juga akan mengurangi ukuran dari *index file* [1], karena dokumen akan direpresentasikan menggunakan stem yang merupakan *index* dari sebuah dokumen [1]. Proses *indexing* dapat mengurangi waktu pemrosesan/pencarian dan kebutuhan memori lebih efisien. Proses *indexing* perlu dilakukan untuk menghindari *linear scanning* karena melakukan *scanning* pada dokumen dalam koleksi setiap kali *query* diberikan adalah tidak praktis bahkan sering tidak mungkin untuk koleksi yang besar, karena proses pencarian akan menjadi sangat lambat [3].

Proses pencarian dokumen yang menggunakan *information retrieval* akan menerima *query* dari pengguna dan mengembalikan dokumen yang relevan dengan *query* pencarian. Hal penting pada sistem *information retrieval* adalah mengurutkan dokumen hasil pencarian berdasarkan relevansi dokumen tersebut dengan *query* yang diberikan [2]. Proses pengurutan tersebut dilakukan dengan memberikan bobot pada dokumen menggunakan fungsi peringkat/pembobotan dokumen, sehingga dokumen yang memiliki tingkat relevansi yang tinggi akan mempunyai nilai bobot yang tinggi. Dokumen yang merupakan hasil

pencarian berdasarkan *query* akan ditampilkan urut berdasarkan nilai bobot dokumen tersebut.

Penelitian ini mengambil sistem pencarian dokumen Tugas Akhir (TA) di Universitas X, di mana terdapat beberapa kriteria pencarian dan sudah menggunakan fungsi peringkat [4]. Namun, pada proses pencarian dokumen TA yang dilakukan tidak adanya proses *indexing* dan belum ada proses *stemming* yang dilakukan untuk *query* maupun data TA. Untuk meningkatkan kinerja dan kehandalan pencarian TA di Universitas X tersebut akan ditambahkan proses *stemming* dan *indexing*, serta pengurutan peringkat dokumen hasil pencarian.

Dokumen TA yang akan dicari menggunakan bahasa Indonesia, maka diperlukan proses *stemming* yang sesuai dengan morfologi bahasa Indonesia. Oleh karena itu pada penelitian ini digunakan *porter stemmer* bahasa Indonesia yang dikembangkan oleh Tala [5]. Dokumen TA akan diindeks menggunakan struktur penyimpanan *inverted index*. Sedangkan untuk mengurutkan dokumen hasil pencarian berdasarkan relevansinya menggunakan fungsi peringkat Okapi BM25.

II. METODOLOGI PENELITIAN

A. *Inverted Index*

Proses *indexing* diperlukan unuk menghindari *linear scanning* pada proses pencarian dokumen dengan *query* [6], di mana hal tersebut dapat mempercepat proses pencarian dari sekumpulan data yang besar [3]. Struktur data pengindeksan *information retrieval* yang terkenal adalah *inverted index*. Untuk setiap kata yang muncul di koleksi dokumen, *inverted index* akan berisi *posting list* atau daftar dokumen yang mempunyai kata tersebut yang berisi informasi kemunculan kata tersebut pada dokumen (dapat berupa frekuensi kata atau posisi) [7].

Pembuatan *inverted index* sebagai berikut, pertama, dibuat kamus *term* (kata) untuk merepresentasikan semua term unik yang ada dalam koleksi dokumen. Frekuensi kemunculan setiap *term* dalam koleksi dokumen juga dapat disimpan. Selanjutnya, diciptakan *posting list* atau *inverted list* untuk setiap *term* di kamus *term* berisi daftar referensi dokumen yang mengandung *term* tersebut. Contoh *inverted index* dapat dilihat pada Gambar 1 untuk *term* "*internet*" terdapat pada dokumen 1, 3, 8, 22, 31, dan 40.

Kamus term	Posting List					
…	…					
internet	1	3	8	22	31	40
…	…					
komputer	1	2	10	30	…	
…	…					
program	2	3	5	…		
…	…					

Gambar 1. Contoh Inverted Index

B. *Porter Stemmer* Bahasa Indonesia

Pada *information retrieval* untuk data tekstual, perlu dilakukan proses *text preprocessing*, salah satu di antaranya proses *stemming*. *Stemming* adalah teknik untuk mendeteksi infleksi dan derivasi yang berbeda dari varian morfologi kata dengan tujuan untuk menguranginya menjadi satu akar (kata dasar) tertentu yang disebut stem [2], berdasarkan asumsi kata yang mempunyai kata dasar yang sama mempunyai arti yang sama [5]. Proses *stemming* merupakan cara untuk meningkatkan kinerja pencarian informasi. Proses *stemming* yang dilakukan waktu pengindeksan, dapat mengurangi ukuran dari *file* indeks.

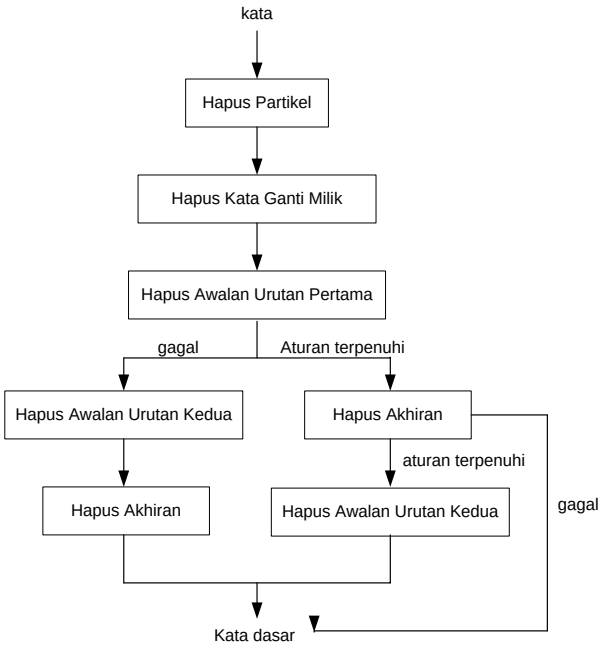
Porter stemmer bahasa Indonesia merupakan *stemmer* berbasis aturan yang dikembangkan dari *porter stemmer* yang dimodifikasi sesuai struktur morfologi kata bahasa Indonesia [5]. Morfologi kata bahasa Indonesia meliputi struktur infleksi dan derivasi. Struktur infleksi adalah struktur paling sederhana yang diekspresikan dengan akhiran yang tidak mempengaruhi atau mengubah arti dasar dari kata dasar [5]. Akhiran infleksi dibagi menjadi dua kelompok: akhiran -lah, -kah, -tah, -pun yang merupakan partikel, dan akhiran -ku,-mu, -nya yang merupakan kata ganti milik. Penambahan akhiran infleksi pada kata dasar tidak mengubah pengejaan kata.

Struktur derivasi bahasa Indonesia terdiri dari awalan, akhiran, dan pasangan kombinasi awalan dan akhiran. Awalan yang paling sering digunakan adalah ber-, di-, ke-, meng-, peng-, per-, ter-. Beberapa awalan seperti ber-, meng-, peng-, per-, ter- dapat muncul dengan berbagai macam bentuk berbeda bergantung dari huruf pertama kata yang dilekati oleh awalan. Pada struktur derivasi, pengejaan kata mungkin dapat berubah ketika awalan ditambahkan. Sedangkan akhiran derivasi adalah -i, -kan, -an. Pelekatan akhiran derivasi tidak mengubah ejaan kata dasar pada kata yang dihasilkan. Struktur derivasi juga meliputi gabungan awalan dan akhiran yang dilekatkan bersamaan pada sebuah kata untuk menghasilkan kata baru. Tidak semua kombinasi gabungan awalan dan akhiran dapat digunakan bersama untuk membentuk sebuah imbuhan, terdapat beberapa awalan dan akhiran yang tidak dapat digunakan bersama. Awalan atau imbuhan dapat ditambahkan ke kata yang sudah diberikan imbuhan atau awalan sebelumnya, di mana menghasilkan awalan ganda. Penambahan akhiran infleksi ke kata yang sudah diberikan imbuhan sebelumnya dapat dilakukan, dan ini merupakan struktur morfologi yang paling kompleks di bahasa Indonesia.

Pada proses *porter stemmer* bahasa Indonesia, di setiap tahap, sebuah awalan/akhiran tertentu dihapus dengan suatu aturan substitusi/pengganti. Aturan substitusi diterapkan ketika sebuah kumpulan kondisi/batasan aturan tersebut dapat dipenuhi benar, di mana menyebabkan penghapusan akhiran dan setelah itu proses berikutnya dilakukan. Jika kondisi dari aturan tertentu tidak dapat terpenuhi, kondisi pada aturan berikutnya akan diuji, sampai sebuah aturan dapat dipenuhi atau sampai aturan habis. Proses akan terus dilakukan untuk semua tahap. Langkah linier dari kumpulan proses ini diharapkan dapat mengurangi kata dengan struktur kompleks di bahasa Indonesia menjadi kata dasar yang

benar. *Porter stemmer* bahasa Indonesia membalik urutan kemunculan imbuhan pada proses pembentukan kata, yaitu akhiran infleksi dihapus pertama kali sebelum imbuhan derivasi. Pada proses pemisahan awalan, akan ada proses koreksi terhadap ejaan kata yang dihasilkan, karena penambahan awalan mungkin akan menyebabkan perubahan ejaan dari kata yang ditambahkan awalan tersebut. Pemisahan awalan selalu sebelum pemisahan akhiran. Kondisi tambahan ditambahkan untuk mengecek kemungkinan sebuah awalan membentuk sebuah kombinasi imbuhan yang legal dengan penghapusan awalan sebelumnya.

Pada *porter stemmer* bahasa Indonesia, kata yang akan dilakukan proses *stem* minimal terdiri dari 2 suku kata. Desain dari proses *Porter Stemmer* Bahasa Indonesia dapat dilihat pada Gambar 2. Langkah-langkah penghapusan imbuhan pada kata dasar yang ada pada desain *porter stemmer* bahasa Indonesia tersebut dilakukan menurut lima kelompok aturan imbuhan dengan memberikan aturan untuk tiap akhiran atau awalan yang ditemukan, akan digantikan dengan apa, serta syarat jumlah minimal suku kata, dan syarat tambahan yang harus dipenuhi sebelum langkah penghapusan dilakukan.



Gambar 2. Proses Porter Stemmer Bahasa Indonesia

Langkah pertama yang dilakukan yaitu menghapus partikel berdasarkan aturan yang terdapat pada Tabel 1. Langkah kedua yang dilakukan yaitu menghapus kata ganti milik berdasarkan aturan yang terdapat pada Tabel 2. Langkah ketiga yang dilakukan yaitu menghapus awalan urutan pertama berdasarkan aturan pada Tabel 3. Pada tabel 3, kolom syarat tambahan, notasi "V..." pada kolom syarat tambahan berarti kata yang telah di-stem harus diawali huruf vokal dan notasi "K..." berarti kata yang telah di-stem harus diawali huruf konsonan.

Tabel 1. Aturan Menghapus Partikel

Akhiran	Pengganti	Jumlah minimal suku kata	Syarat tambahan	Contoh
kah	NULL	2	NULL	bukukah → buku
lah	NULL	2	NULL	inilah → ini
pun	NULL	2	NULL	bukupun → buku

Tabel 2. Aturan Menghapus Kata Ganti Milik

Akhiran	Pengganti	Jumlah minimal suku kata	Syarat tambahan	Contoh
Ku	NULL	2	NULL	bukuku → buku
mu	NULL	2	NULL	bukumu → buku
nya	NULL	2	NULL	bukunya → buku

Tabel 3. Aturan Menghapus Awalan Urutan Pertama

Awalan	Penggan ti	Jumlah minimal suku kata	Syarat tambahan	Contoh
Meng	NULL	2	NULL	mengukur → ukur
Meny	*	2	V…*	menyapu → sapu
Men	NULL	2	NULL	menduga → duga
Mem	p	2	V…	memilah → pilah
Mem	NULL	2	NULL	membaca → baca
Me	NULL	2	NULL	merusak → rusak
Peng	NULL	2	NULL	pengukur → ukur
Peny	*	2	V…	penyapu → sapu
Pen	NULL	2	NULL	penduga → duga
Pem	p	2	V…	pemilah → pilah
Pem	NULL	2	NULL	pembaca → baca
di	NULL	2	NULL	diukur→ ukur
ter	NULL	2	NULL	tersapu→ sapu
kekasih	NULL	2	NULL	kekasih→ kasih

Jika tidak ada awalan urutan pertama yang dihapus, langkah keempat yang dilakukan adalah menghapus awalan urutan kedua berdasarkan aturan yang terdapat pada Tabel 4. Diikuti dengan menghapus akhiran berdasarkan aturan yang terdapat pada Tabel 5. Jika ada awalan urutan pertama yang dihapus, langkah keempat yang dilakukan adalah menghapus akhiran yang diikuti dengan menghapus awalan urutan kedua jika ada akhiran yang dihapus. Variabel k adalah parameter *tuning* positif yang menyesuaikan *scaling term frequency* dokumen; k=0 dapat disamakan dengan *binary model* (tidak ada *term frequency*). Variabel b (0≤b≤1) adalah parameter *tuning* lain yang menentukan *scaling* oleh panjang dokumen: b=0 berarti tidak ada normalisasi untuk panjang dokumen [6].

Tabel 4. Aturan Menghapus Awalan Urutan Kedua

Awalan	Penggan ti	Jumlah minimal suku kata	Syarat tambahan	Contoh
ber	NULL	2	NULL	berlari → lari
bel	NULL	2	ajar	belajar → ajar
be	NULL	2	K*er	bekerja → kerja
per	NULL	2	NULL	perjelas → jelas
pel	NULL	2	ajar	pelajar → ajar
pe	NULL	2	NULL	pekerja → kerja

Tabel 5. Aturan Menghapus Akhiran

Akhiran	Penggan ti	Jumlah minimal suku kata	Syarat tambahan	Contoh
kan	NULL	2	awalan ∉{ke, peng}	tarikkan → tarik
an	NULL	2	awalan ∉{di, meng, ter}	makanan → makan
i	NULL	2	V K …c1c2, c1≠s.c2≠i dan awalan ∉ {ber, ke, peng}	tandai→ tanda

C. Okapi BM25

Okapi BM25 merupakan fungsi peringkat yang dibuat oleh Stephen Robertson dan Karen Sparck Jones [8]. Sebuah fungsi peringkat mengambil sebuah dokumen dan sebuah *query*, kemudian mengembalikan sebuah nilai numerik sebagai bobot atau ukuran relevansi dokumen dengan *query*. Dokumen yang mempunyai nilai fungsi tertinggi diasumsikan sebagai dokumen yang paling relevan dengan *query* (*keyword*) yang diberikan. Okapi BM25 menentukan peringkat dokumen terhadap kata kunci *query* yang diinputkan berdasarkan tiga faktor, yaitu TF (*term frequency*), IDF (*inverse document frequency of term*), dan panjang dokumen. Fungsi Okapi BM25 didefinisikan sebagai berikut:

$$BM25(d_j, q_{1:N}) = \sum_{i=1}^{N} IDF(q_i) \frac{TF(q_i, d_j)(k+1)}{TF(q_i, d_j) + k(1 - b + b\frac{|d_j|}{L})} \quad (1)$$

Fungsi IDF sebagai berikut:

$$IDF = \log \frac{N - DF(q_i) + 0.5}{DF(q_i) + 0.5} \quad (2)$$

di mana q_i adalah kata yang dicari, $|d_j|$ adalah panjang dokumen d_j (banyaknya kata dalam dokumen d_j), $TF(q_i,d_j)$ adalah jumlah kemunculan kata q_i dalam sebuah dokumen d_j, $IDF(q_i)$, yaitu nilai invers dari total dokumen yang berisi kata q_i, L adalah panjang dokumen dari N dokumen, k adalah parameter *term frequency* (pada penelitian ini digunakan k=2), b = parameter panjang dokumen (0≤b≤1), IDF(qi) adalah nilai invers dari total dokumen yang mengandung kata qi, N adalah jumlah seluruh dokumen, DF(qi) merupakan *document frequency* (jumlah dokumen yang mengandung kata qi). Variabel k adalah parameter *tuning* positif yang menyesuaikan *scaling term frequency* dokumen; k=0 dapat disamakan dengan binary model (tidak ada *term frequency*). Variabel b (0≤b≤1) adalah parameter *tuning* lain yang menentukan *scaling* oleh panjang dokumen: b=0 berarti tidak ada normalisasi untuk panjang dokumen [6].

III. HASIL DAN PEMBAHASAN

Proses yang dilakukan pada sistem pencarian dokumen Tugas Akhir berdasarkan *query* sebagai berikut:

1. *Text Preprocessing*
   Text preprocessing dilakukan untuk mengubah teks menjadi bentuk yang siap untuk diproses lebih lanjut serta mengurangi jumlah *term* yang akan diproses. Langkah pertama yaitu *case folding* di mana semua karakter dalam dokumen diubah menjadi huruf kecil dan karakter selain huruf dihilangkan. Langkah selanjutnya yaitu *lexical analysis* (*tokenizing*) yang membaca karakter input dan menghasilkan output berupa *token* (kata/*term*) dan membuang spasi, *tab*, *newline*, dan karakter-karakter lain yang tak berguna. Setelah itu dilakukan *stopword removal* yang menghilangkan kata-kata yang dianggap tidak penting dari daftar kata hasil *tokenizing*. Kemudian dilakukan proses *stemming* menggunakan *porter stemmer* bahasa Indonesia untuk mengubah kata berimbuhan menjadi kata dasarnya. Proses *stemming* akan dilakukan pada judul, abstrak, dan kata kunci dokumen tugas akhir yang kemudian disimpan di *inverted index*. Proses *stemming* juga dilakukan pada *query* yang diinputkan *user*.
2. Pembuatan *inverted index*

Pada proses pembuatan *inverted index* diambil judul, abstrak, dan kata kunci dokumen TA dari *database* dan dilakukan proses *text preprocessing* untuk dokumen tersebut. Kemudian untuk tiap kata (*term*) akan dipetakan ke dokumennya masing-masing. Kata kemudian diurutkan berdasarkan abjad; kemudian kata yang muncul lebih dari sekali dalam sebuah dokumen digabungkan dan frekuensi kemunculannya dalam dokumen tersebut disimpan. Terakhir, dibuat *dictionary* dengan posting *list*-nya yaitu tabel kata, id dokumen yang mengandung kata tersebut, dan frekuensi kemunculan kata tersebut dalam dokumen tersebut; hasilnya adalah *inverted index* yang akan disimpan di *database*.

3. Pembobotan dokumen
   Pada proses pembobotan dokumen, sistem menerima *input* berupa *query* (*keyword* pencarian) dari *user*. *Query* kemudian dibandingkan dengan data dokumen TA dari *inverted index* menggunakan fungsi Okapi BM25. *Output* yang diberikan yaitu daftar dokumen yang telah diurutkan sesuai relevansinya sebagai hasil pencarian.

Dataset yang digunakan untuk melakukan evaluasi ini adalah data tugas akhir mahasiswa Teknik Informatika Universitas X yang berjumlah 59 data [4]. Uji coba dilakukan untuk menguji hasil peringkat pencarian tugas akhir dengan menggunakan beberapa *query*. Sebagai contoh jika query yang diinputkan adalah "pakar bayi", maka menghasilkan 6 dokumen tugas akhir dari 59 dokumen yang pada judul, abstrak, ataupun kata kuncinya terdapat kata pakar atau bayi dengan urutan hasil pencarian berdasarkan nilai peringkat menggunakan algoritma OKAPI BM25. Hasil perhitungan idf *query* "pakar bayi" dapat dilihat pada Tabel 6. Sedangkan hasil perhitungan OKAPI BM25 untuk pencarian dengan *query* "pakar bayi" dapat dilihat pada Tabel 7.

Pada Tabel 8 dapat dilihat bahwa hasil dokumen pencarian TA sudah diurutkan berdasarkan nilai peringkat OKAPI BM 25. *Query* "pakar bayi" menghasilkan 6 dokumen, di mana hanya pada dokumen no urut 1 yang mempunyai term pakar sekaligus bayi pada judul, abstrak, dan kata kunci. Dokumen lain hanya mempunyai salah satu dari term pencarian, pada dokumen no urut 2 hanya mempunyai *term* bayi, sedangkan dokumen no urut 3 sampai dengan 6, hanya mempunyai *term* pakar. Oleh karena itu, nilai peringkat dokumen no 1 paling besar di antara dokumen lain, dan selisih nilai peringkat dengan dokumen yang lain sangat besar.

Uji coba juga dilakukan dengan melakukan perbandingan sistem yang menggunakan *stemmer* dan sistem yang tidak menggunakan *stemmer*. Uji coba dilakukan dengan membandingkan jumlah baris data pada tabel penyimpanan untuk sistem yang menggunakan *stemmer* dan atau melakukan proses *stopword removal* dengan sistem yang tidak menggunakan kedua proses tersebut. Perhitungan jumlah baris data dilakukan pada tabel *dictionary database inverted index* yang memiliki kolom *term* dan dokumen sebagai *primary key*.

Dari 59 data TA, sistem yang menggunakan *stemmer* dan melakukan *stopword removal* memiliki 3560 baris data pada tabel *dictionary*. Sementara sistem yang tidak menggunakan *stemmer* memiliki 3802 baris data. Sistem yang tidak menggunakan *stemmer* dan tidak melakukan *stopword removal* memiliki 6412 baris data. Tabel perbandingan jumlah baris data pada *inverted index* ketiga sistem tersebut dapat dilihat pada Tabel 7. Jumlah data yang lebih sedikit pada sistem dengan *stemmer* dan *stopword removal* membuktikan bahwa penggunaan *stemmer* dan adanya proses *stopword removal* dapat membantu untuk mengurangi besar tabel penyimpanan.

Tabel 6. Hasil Perhitungan IDF Untuk Term Pakar dan Term Bayi

Term Query	**DF**	**Idf**
Pakar	5	2.47
Bayi	2	3.38

Tabel 7. Perbandingan Jumlah Baris Data yang Dihasilkan Pada Inverted Index

Keterangan Sistem	**Jumlah Baris Data Pada Tabel *Inverted Index***
Sistem yang dibuat (menggunakan *stemmer* dan melakukan *stopword removal*)	3560
Sistem tidak menggunakan *stemmer*	3802
Sistem yang tidak menggunakan *stemmer* dan tidak melakukan *stopword removal*	6412

Uji coba juga dilakukan untuk membandingkan kecepatan dilakukan pada sistem yang telah dibuat menggunakan *inverted index* dengan sistem yang tidak menggunakan *inverted index* untuk pencarian dokumen TA. Uji coba ini dilakukan dengan cara menginputkan *query* (*keyword* pencarian) dengan jumlah kata yang bervariasi untuk membandingkan kecepatan respon dari kedua sistem. Perhitungan waktu eksekusi pencarian dilakukan menggunakan *benchmarking class code igniter*. Waktu eksekusi pencarian pada sistem yang dibuat dan sistem tanpa *inverted index* dapat dilihat pada Tabel 9.

Dari hasil waktu eksekusi yang didapatkan terlihat bahwa sistem tanpa *inverted index* memerlukan waktu yang sangat lama untuk melakukan pencarian untuk *query* dengan jumlah kata yang semakin banyak. Hal ini menunjukkan bahwa sistem juga akan memerlukan waktu yang sangat lama untuk melakukan pencarian untuk koleksi dengan jumlah dokumen yang semakin banyak. Hal ini dikarenakan tidak adanya *inverted index* mengharuskan sistem membaca setiap dokumen secara linear untuk mendapatkan informasi yang dibutuhkan dalam perhitungan nilai Okapi BM25 setiap kali query diberikan.

Tabel 8. Hasil Perhitungan Nilai Okapi BM25 Pada Dokumen Hasil Pencarian Untuk Query “Pakar Bayi”

No Urut	Judul dokumen	Panjang Dokumen	*Query*	TF	Nilai	Nilai dokumen
1	Pembuatan Sistem **Pakar** untuk Mendiagnosa Masalah Gangguan Tidur pada **Bayi**	137	Pakar	9	5.99	14.71
			Bayi	13	8.72	
2	Pembuatan Aplikasi Web pada Toko Perlengkapan **Bayi** untuk Memperluas Pelayanan kepada Pelanggan	119	Pakar	0	0	6.2
			Bayi	3	6.2	
3	Pembuatan Aplikasi Sistem **Pakar** untuk Membantu Mendiagnosa Penyakit dan Hama pada Tanaman Anggrek berbasis Web	166	Pakar	12	6.14	6.14
			Bayi	0	0	
4	Pembuatan Program Sistem **Pakar** untuk Melakukan Diagnosa Awal Penyakit Sinusitis Berbasis Web	164	Pakar	10	5.95	5.95
			Bayi	0	0	
5	Sistem **Pakar** untuk Diagnosa Kerusakan Mobil Berbahan Bakar Solar Berbasis Web	131	Pakar	8	5.89	5.89
			Bayi	0	0	
6	Pembuatan Aplikasi Sistem **Pakar** untuk Membantu Menentukan Pola dan Teknik Latihan *Fitness*	94	Pakar	6	5.83	5.83
			Bayi	0	0	

Tabel 9. Perbandingan Lama Waktu Pencarian

Jumlah Kata Dalam *Query*	Waktu Ekseksi (Detik)	
	Sistem Yang Dibuat (Dengan *Inverted Index*)	Sistem Tanpa *Inverted Index*
1	2,1488	1,0348
5	2,1388	2,2256
10	2,1501	2,8606
20	2,1034	5,3801
32	2,1256	8,3232
100	2,2418	31,5562
152	2,2036	50,3019
246	2.2306	75,1968

. Dengan demikian dapat disimpulkan bahwa adanya *inverted index* pada sistem yang telah dibuat dapat membantu mengurangi waktu yang dibutuhkan untuk melakukan pencarian secara signifikan.

Proses *stemming* memungkinkan ditemukannya lebih banyak dokumen yang memiliki kemungkinan relevansi dengan kebutuhan informasi yang diinginkan *user* serta hasil pencarian yang lebih fleksibel dibandingkan sistem yang tidak menggunakan *stemmer*. Uji coba perbandingan hasil pencarian ini dilakukan dengan menginputkan sebuah *keyword* pencarian dan kemudian membandingkan hasil pencarian dari sistem yang menggunakan *stemmer* dengan sistem yang tidak menggunakan *stemmer*.

Keyword yang diinputkan sebagai contoh untuk uji coba ini adalah "pemasaran". Pada sistem yang tidak menggunakan *stemmer* pencarian untuk *keyword* "pemasaran" tidak menemukan hasil dan akan menampilkan pesan *no result*. Sementara pada sistem yang menggunakan *stemmer* ditemukan dua dokumen tugas akhir untuk *keyword* "pemasaran".

Pada sistem yang dibuat menggunakan *stemmer*, kata "pemasaran" akan di-*stem* ke kata dasarnya menjadi "pasar". Oleh karena itu, sistem dapat menemukan kata "pasar" yang terdapat pada dokumen tugas akhir urutan pertama. Sistem juga dapat menemukan kata "memasarkan" dan "pemasarannya" yang terdapat pada dokumen tugas akhir kedua karena data dokumen tugas akhir juga telah melalui proses *stemming* sebelum disimpan di *database inverted index*, sehingga kata "memasarkan" dan "pemasarannya" juga telah di-*stem* ke kata dasarnya menjadi "pasar".

Untuk mengetahui performa *stemmer* yang telah dibuat dengan algoritma *Porter Stemmer* Bahasa Indonesia dilakukan evaluasi hasil stemming yang dilakukan secara manual dengan melakukan pengamatan secara langsung terhadap hasil *stemming*. Evaluasi hasil stemming pertama dilakukan pada varian morfologi kata "simpan", yaitu "penyimpanan", "menyimpan", "tersimpan", dan "simpanan"; di mana sistem sudah dapat mengembalikan semua kata tersebut ke bentuk *stem* yang sama yaitu "simpan".

Evaluasi hasil *stemming* kedua dilakukan pada varian morfologi kata "kembali", yaitu "pengembalian" dan "mengembalikan". Di mana *stemmer* akan melakukan *stem* pada kata "kembali" menjadi "kembali", namun kata "pengembalian" dan "mengembalikan" akan di-*stem* menjadi "embali". Sehingga kata "kembali", "pengembalian", dan "mengembalikan" yang merupakan kelompok semantik yang sama di-*stem* menjadi bentuk yang berbeda (terjadi *understemming*).

Evaluasi hasil *stemming* ketiga dilakukan pada kata "majalah". Di mana *stemmer* akan melakukan stem pada kata "majalah" menjadi "maja". Kata "maja" memiliki arti yang berbeda dan merupakan kelompok semantik yang berbeda dari kata "majalah". Sehingga pada kasus ini sistem

melakukan kesalahan *overstemming* pada kata "majalah" dan "maja".

Evaluasi hasil *stemming* keempat dilakukan pada kata "*retrieval*" dan "*retrieving*". Di mana *stemmer* tidak dapat mengembalikan kedua kata ini menjadi kata dasarnya yaitu "*retrieve*". Hal ini dikarenakan algoritma *Porter Stemmer* Bahasa Indonesia yang dibuat oleh Tala tidak memiliki aturan untuk varian morfologi dalam Bahasa Inggris. Kesalahan-kesalahan *understemming* dan *overstemming* yang terdapat pada *stemmer* dapat dikurangi dengan menggunakan algoritma *stemmer* yang menggunakan suatu kamus kata dasar. Kemudian untuk dapat melakukan *stem* untuk kata-kata dalam Bahasa Inggris perlu digunakan algoritma *stemmer* khusus untuk Bahasa Inggris.

Sedangkan untuk menilai efektivitas suatu sistem IR yaitu kualitas hasil pencarian, dua ukuran yang paling sering digunakan adalah *precision* dan *recall*. *Precision* (P) adalah rasio jumlah dokumen relevan yang ditemukan dengan total jumlah dokumen yang ditemukan. *Recall* (R) adalah rasio jumlah dokumen relevan yang ditemukan dengan total jumlah dokumen dalam koleksi yang dianggap relevan. Perhitungan *precision* dan *recall* sebagai berikut:

$$\text{Precision} = \frac{\text{jumlah dokumen relevan yang terambil}}{\text{jumlah dokumen yang terambil}} \quad (3)$$

$$\text{Recall} = \frac{\text{jumlah dokumen relevan yang terambil}}{\text{jumlah dokumen relevan}} \quad (4)$$

Digunakan 60 *query* untuk perhitungan nilai *precision* dan *recall* untuk setiap *query* tersebut. 30 dari *query* merupakan beberapa kata dasar dan variasi kata yang menggunakan kata dasar tersebut. Kata dasar yang digunakan meliputi: ajar, program, buat, putus, hubung, angkat, kenal, dan dukung. Sedangkan 30 *query* yang lainnya merupakan *query* berupa kata yang tidak mempunyai variasi kata dasar yang sama atau *query* yang terdiri dari 2 kata. Setelah itu, dihitung nilai *precision* rata-rata dan *recall* rata-rata dengan menghitung total dokumen yang dihasilkan dari proses pencarian tersebut, seperti tampak pada Tabel 10.

Tabel 10. Dokumen Hasil *Query* Uji Coba

Keterangan	Jumlah
Total dokumen relevan yang terambil	643
Total dokumen yang terambil	652
Total dokumen relevan	643
Rata-rata precision	**0.986**
Rata-rata recall	**1**

Dari hasil perhitungan yang didapatkan, terlihat bahwa sistem IR memiliki nilai *recall* yang sangat baik yaitu 1 dan nilai *precision* yang baik yaitu 0.986. Pada *query* yang memiliki nilai *precision* kurang dari precision rata-rata, dokumen yang relevan tetap berada di urutan atas dan dokumen yang tidak relevan di urutan bawah, sehingga kurangnya precision ini juga tidak akan terlalu mengganggu *user* untuk menemukan TA yang sesuai dengan kebutuhannya.

Rendahnya *precision* pada beberapa *query* dalam evaluasi ini disebabkan oleh sistem menampilkan semua dokumen yang sesuai untuk tiap kata dalam *query*, padahal seringkali *query* yang terdiri dari beberapa kata memiliki kebutuhan informasi yang lebih spesifik daripada masing-masing kata dalam *query* tersebut. Penggunaan *stemmer* pada sistem ini membantu meningkatkan *recall* karena dokumen yang mengandung bentuk morfologi lain dari suatu kata juga akan terambil. Namun penggunaan *stemmer* juga dapat mengurangi *precision* karena bentuk lain dari kata tersebut bisa jadi memiliki arti lain dan tidak sesuai dengan kebutuhan informasi yang diinginkan *user*.

IV. KESIMPULAN

Hasil pencarian dapat ditampilkan sesuai urutan relevansi dokumen hasil pencarian dengan query yang diberikan dengan menggunakan algoritma Okapi BM25. Penggunaan *inverted index* pada sistem dapat mempercepat waktu respon sistem untuk melakukan proses pencarian secara signifikan. Penggunaan *stemmer* dapat membantu menemukan lebih banyak dokumen yang memiliki kemungkinan relevansi dengan kebutuhan informasi yang diinginkan *user*. Penggunaan *stemmer* dan adanya proses *stopword removal* dapat membantu untuk mengurangi besar tabel penyimpanan yang juga akan mempercepat proses pengambilan data dokumen yang mengandung suatu *term* dari *inverted index*.

Pengembangan sistem dapat dilakukan untuk mengurangi kesalahan pencarian berupa *understemming* dan *overstemming* yang terdapat pada *stemmer* dapat digunakan algoritma *stemmer* yang menggunakan suatu kamus kata dasar. Selain itu, untuk dapat melakukan *stem* untuk kata-kata dalam bahasa Inggris perlu digunakan algoritma *stemmer* khusus untuk bahasa Inggris.

REFERENSI

[1] Karaa, W.B.A. (2013). *A New Stemmer to Improve Information Retrieval*. International Journal of Network Security & Its Applications (IJNSA).

[2] He, B., Ounis, I. (2005). *Term Frequency Normalisation Tuning for BM25 and DFR Models*, ECIR:Springer.

[3] Ceri, S. dkk. (2013). *Web Information Retrieval*. Milan, Italy: Springer-Verlag Berlin Heidelberg.

[4] Tjandra, E., Widiasri, M. (2015). *Sistem Repositori Tugas Akhir Mahasiswa dengan Fungsi Peringkat Okapi BM25*. Surabaya: Universitas Airlangga.

[5] Tala, F. Z. (2003). *A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia*. [Online] Netherlands: Universiteit van Amsterdam. Available: https://www.illc.uva.nl/Research/Publications/Reports/MoL-2003-02.text.pdf.

[6] Manning, C.D., Raghavan, P., Schütze, H. (2008). *An introduction to information retrieval*. England: Cambridge University Press.

[7] Catena, M., Macdonal C., dan Ounis, I. (2014). *On Inverted Index Compression for Search Engine Efficiency*, Switzerland: Springer.

[8] Russell, S. dan Norvig, P. (2009). *Artificial Intelligence : a Modern Approach*. 3rd ed. Upper Saddle River, NJ: Prentice Hall.