IMPLEMENTASI MULTITHREADING UNTUK MENINGKATKAN KINERJA INFORMATION RETRIEVAL DENGAN METODE GVSM

Jasman Pardede[1]

Abstract

GVSM is one of the models of IR systems. GVSM divided into two processes i.e. preprocessing process namely reading the text (*.pdf, *.doc,*.docx), tokenization, filtration, stemming, and parse the query and the process of calculating the relevance of the document that has been done preprocessing with user query to get the value of similarity. In general, to determine of the effectiveness of IR application is determined by two factors, namely recall and precision [7,8,10]. According to Järvelin and Ingwersen, the effectiveness of IR application is not only determined by the recall and precision, but rather the ability of IR applications in helping users to complete the search more effective and efficient. The effectiveness of an IR application is determined by recall, precision, time requirements, and reporting documents are presented to the user [7]. To improve the performance of IR applications in time requirements is implemented multithreading in the preprocessing process, i.e. stage 1 to stage 4 GVSM method. The result showed that the method GVSM able to rediscover relevant documents with 100% the precision value, to get the recall value which is equal to the same documents collection either with or without multithreading, but can save processing time by over 50 %.

Kata kunci : *GVSM, multithreading,recall, precision, effectiveness.*

1. Pendahuluan

Generalized Vector Space Model (GVSM) merupakan salah satu model sistem information retrieval (IR) [9,11,12]. Proses yang terjadi pada GVSM terbagi menjadi dua proses yaitu proses *preprocessing* dan proses menghitung relevansi antara kumpulan dokumen yang telah di-*preprocess* dengan *query* yang diinginkan pengguna berdasarkan nilai similaritasnya. Proses *preprocessing* yaitu *reading text* (*.pdf,*.doc,*.docx), *tokenization*, *filtration, stemming* dan *parse query*.

[1] Jurusan Teknik Informatika, Fakultas Teknik Industri, Itenas Bandung
Jln. PKH. Hasan Mustapa No.23 Bandung 40124
[1] jasman@itenas.ac.id, pardedc_js@yahoo.com

Di dalam aplikasi IR hanya mendapatkan dokumen yang relevan tidaklah cukup, tetapi dokumen yang telah didapatkan tersebut harus dapat ditampilkan terurut dari dokumen yang memiliki tingkat relevansi tinggi ke tingkat relevansi yang lebih rendah. Penyusunan dokumen tersebut disebut sebagai perangkingan dokumen [13].

Pada umumnya alat ukur penentuan keefektifan suatu aplikasi IR ditentukan oleh dua faktor, yaitu *recall* dan *precision* [7,8,10]. Sedangkan menurut Järvelin and Ingwersen, keefektifan aplikasi IR bukan hanya ditentukan oleh *recall* dan *precision* tetapi lebih kepada kemampuan aplikasi IR dalam membantu pengguna untuk menyelesaikan pencarian yang lebih efektif dan efisien. Keefektifan suatu aplikasi IR ditetukan oleh *recall*, *precision*, kebutuhan waktu, dan pelaporan dokumen yang disajikan kepada pengguna [7]. Untuk meningkatkan kinerja aplikasi IR dalam kebutuhan waktu peneliti mengimplementasikan multithreading sehingga mampu memproses banyak thread pada waktu yang sama pada kumpulan dokumen.

Berdasarkan uraian latar belakang di atas, maka peneliti menemukan rumusan masalah seperti berikut :

1. Bagaimana sistem membaca dokumen berformat *.doc, *docx, dan *.pdf.
2. Bagaimana mengimplementasikan *multi-threading* pada proses *preprocessing* dapat bekerja pada sistem.
3. Bagaimana memperoleh dokumen yang relevan dan sudah terurut sesuai dengan *query* yang dimasukan pengguna dengan menggunakan metode *GVSM*.

Adapun tujuan dari penelitian ini adalah melakukan analisis dan implementasi *multithreading* untuk meningkatkan kinerja IR dengan metode GVSM.

Metodologi yang dipakai adalah prototype, yaitu pembuatan prototype *preprocessing* dengan proses *stemming* mengikuti tahapan algoritma Nazief dan Adriani, menghitung index term, mengubah dokumen dan query menjadi vektor, dan pengurutan dokumen sesuai dengan nilai similaritas.

2. Tinjauan Pustaka

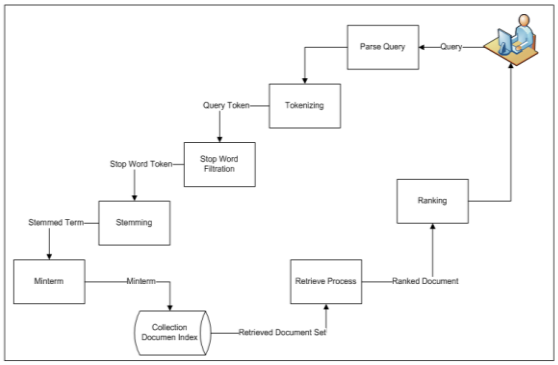
Pada subbab ini akan membahas tentang pengertian Information Retrieval, metode GVSM, Multithreading dan algoritma Nazief dan Adriani

2.1 Information Retrieval

IR system atau sistem temu balik informasi merupakan bagian dari ilmu komputer tentang pengambilan informasi dari dokumen-dokumen yang didasarkan pada isi dan konteks dari dokumen-dokumen itu sendiri[1, 13]. Menurut Gerald Kowalski, sistem temu balik informasi adalah suatu sistem yang mampu melakukan penyimpanan, pencarian, dan pemeliharaan informasi. Informasi dalam konteks ini dapat terdiri dari teks (termasuk data numerik dan tanggal), gambar, audio, video, dan objek multimedia lainnya [3]. Definisi IR adalah bagaimana menemukan suatu dokumen dari dokumen-dokumen tidak terstruktur yang memberikan informasi yang dibutuhkan dari koleksi dokumen yang sangat besar yang tersimpan dalam komputer [4].

Query dalam information retrieval merupakan sebuah formula yang digunakan untuk mencari informasi yang dibutuhkan oleh pengguna. Sebuah query merupakan suatu *keywords* (kata kunci) dan dokumen yang mengandung *keywords* merupakan dokumen yang dicari dalam information retrieval.

Model IR adalah model yang digunakan untuk melakukan pencocokan antara term-term dari query dengan term-term dalam *document collection* (folder file). Model yang terdapat dalam information retrieval terbagi dalam 3 model besar, yaitu *Set-theoritic models, Algebraic model,* dan *Probabilistic model* [9, 11, 12]. Skema proses pencarian dokumen berdasarkan query pengguna seperti yang dinyatakan pada Gambar 1.



Gambar 1. Skema proses pencarian dokumen

2.2 Metode GVSM

Generalized Vector Space Model (GVSM) merupakan perluasan dari Vector Space Model (VSM) yaitu dengan menambahkan jenis informasi tambahan, disamping term, dalam merepresentasikan dokumen. IR dengan GVSM merepresentasikan dokumen dengan similaritas vektor terhadap semua dokumen yang ada [9,11,12].

Pada tahun 1985, Wong et al. menyajikan suatu alternatif terhadap IR VSM, yang disebut GVSM. Deskripsi ringkas mengenai GVSM diberikan oleh Carbonell dkk. Asumsikan term dari VSM adalah *liniearly independent*; GVSM menghindari pengaksumsian dengan penggunaan dokumen-dokumen sebagai dasar ruang vector dari pada term. Dalam "*Dual Space*" suatu dokumen direpresentasikan oleh suatu vector dimana dimensinya merujuk terhadap dokumen.

Menurut Baeza, metode GVSM dalam pecarian dokumen mengikuti tahapan sebagai berikut [11] :

1. Membuang kata depan dan kata penghubung.
2. Menggunakan stemmer pada kumpulan dokumen dan query, yaitu aplikasi yang digunakan untuk menghilangkan imbuhan (awalan, akhiran).
3. Menentukan minterm untuk menentukan kemungkinan pola frekuensi kata. Panjang minterm ini didasarkan pada banyak kata yang dimasukan pada query. Kemudian diubah menjadi vektor ortogonal sesuai dengan pola minterm yang muncul.
4. Menghitung banyaknya frekuensi atau kemunculan kata dalam kumpulan dokumen yang sesuai dengan query.
5. Menghitung index term yang dapat dinyatakan dengan :

$$\overrightarrow{Ki} = \frac{\sum_{\forall r, gi(Mr)=1} Ci,r.\overrightarrow{Mr}}{\sqrt{\sum_{\forall r, gi(Mr)=1} Ci,r^2}} \quad \text{.....(1)}$$

Dimana :

$\overrightarrow{Ki}$: Index term ke-i

$\overrightarrow{Mr}$: Vektor ortogonal sesuai pola minterm yang terpakai

Ci,r : Faktor korelasi antara Index term ke-i dengan minterm r

Menghitung faktor korelasi sebagai berikut :

$$C_{i,r} = \sum_{d_j | g_i(\overrightarrow{dj}) = g_i(M_r)} W_{i,j} \quad \text{.....(2)}$$

Dimana :

$W_{i,j}$: Berat *index term* i pada dokumen j

$g_i(M_r)$: Bobot *index term* K_i dalam *minterm* M_r

6. Mengubah dokumen dan query menjadi vektor :

$$\overrightarrow{d_j} = \sum_{i=1}^{n} W_{i,j} \text{x} \overrightarrow{K_i} \quad \text{.....(3)}$$

$$\vec{q} = \sum_{i=1}^{n} q_i \text{x} \overrightarrow{K_i} \quad \text{.....(4)}$$

Dimana :

$\overrightarrow{d_j}$: Vektor dokumen ke-j

$\vec{q}$: Vektor *query*

$W_{i,j}$: Berat *index term* i pada dokumen j

q_i : Berat *index term* pada *query* i

n : Jumlah index term

7. Mengurutkan dokumen berdasarkan similaritas, dengan menghitung perkalian vektor :

$$\text{Sim}\left(\vec{d_j}.\vec{q}\right) = \frac{\vec{d_j}.\vec{q}}{|\vec{d_j}||\vec{q}|} \quad \text{.....(5)}$$

2.3 *Multithreading*

Untuk memahami multithreading, perlu memahami konsep proses dan *thread*. Sebuah proses adalah suatu program dalam eksekusi. Proses dapat dibagi kedalam sejumlah unit-unit terpisah yang dikenal sebagai *thread*. *Multithreading* adalah kemampuan dari suatu program atau sistem operasi untuk mengatur lebih dari satu pengguna pada waktu yang bersamaan dan menangani banyak permintaan pengguna tanpa melakukan penggandaan atau peng-*copy*-an program yang berjalan pada komputer. Dengan kata lain, *Multithreading* adalah kemampuan memproses banyak *thread* pada waktu yang sama dari pada banyak proses. Berdasarkan asal katanya *multithreading* dapat diterjemahkan sebagai pengontrolan banyak *thread* (*multiple thread*) dimana setiap *thread* berjalan secara terpisah-pisah. Adapun keuntungan *multithreading* adalah [6]:

1. Meningkatkan responsif aplikasi
2. Penggunaan multiprosesos yang efisien
3. Meningkatkan Struktur program
4. Penggunaan Sumber daya sistem yang lebih kecil

2.4 *Algoritma* Nazief dan Adriani

Algoritma Nazief dan Adriani adalah algoritma *stemming* untuk bahasa Indonesia. Algoritma ini dideskripsikan pertama kali dalam *unpublished technical report* di Universitas Indonesia. Algoritma ini menggunakan beberapa aturan morfologi untuk menghilangkan affiks (awalan, imbuhan, dll) dari sebuah kata dan kemudian mencocokannya dalam *database* akar kata (kata dasar / *root word*).

Dasar utama algoritma ini adalah daftar akar kata. Langkah pertama yang dilakukan adalah mengumpulkan daftar akar kata dalam bahasa Indonesia. Semakin lengkap daftarnya, semakin tinggi akurasi algoritma ini. Detail algoritma Nazief dan Adriani dapat dilihat pada[5].

2.5 Pengukuran Efektivitas Information Retrieval

Menurut Järvelin and Ingwersen, efektivitas aplikasi IR bukan hanya ditentukan oleh *recall* dan *precision* tetapi lebih kepada kemampuan aplikasi IR dalam membantu pengguna untuk menyelesaikan pencarian yang lebih efektif dan efisien. Keefektifan suatu aplikasi IR ditetukan oleh *recall*, *precision*, kebutuhan waktu, dan pelaporan dokumen yang disajikan kepada pengguna[2]. *Recall* adalah proporsi jumlah dokumen yang dapat ditemukan-kembali oleh sebuah aplikasi pencarian. *Recall* adalah perbandingan antara jumlah dokumen yang relevan yang ditemukan dengan jumlah semua dokumen yang ada pada *document collection*. *Precision* adalah kecocokan atau kesesuaian antara permintaan dengan jawaban terhadap permintaan tersebut. *Precision* adalah perbandingan antara jumlah dokumen yang ditemukan dengan jumlah semua dokumen yang relevan dan tidak relevan yang ditemukan [8].

3. Hasil Penelitian

Pada subbab berikut ini akan membahas tentang analisis GVSM, analisis sistem IR, Analisis Sistem Aplikasi dan perancangan perangkat lunak.

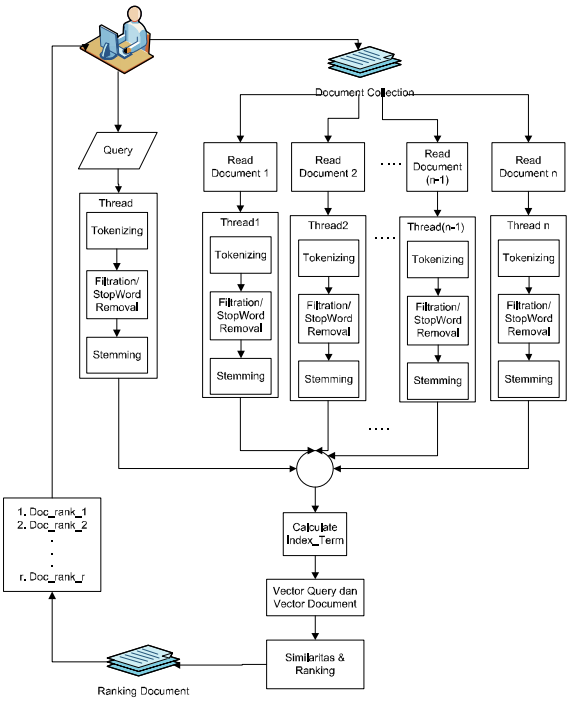
3.1 Analisis GVSM

Berdasarkan subbab 2.2 diperoleh bahwa penerapan algoritma GVSM dalam pecarian dokumen menggunakan empat tahap yang berulang-ulang pada setiap dokumen yang terdapat pada *document collection*, yaitu tahap 1 sampai tahap 4. Dengan menggunakan prinsip *multithreading* yang dibahas pada subbab 2.3, pada penelitian ini akan mengimplementasikan satu *Thread* untuk melakukan tahap 1 sampai tahap 4 pada setiap dokumen. Banyaknya *thread* yang berjalan secara bersamaan adalah sebanyak *document collection*. Hasil tahap 1 sampai tahap 4 dari setiap dokumen disimpan dalam suatu array kemudian digabungkan untuk melakukan perhitungan tahapan 5, 6 dan 7 sehingga akan diperoleh dokumen yang sudah terurut sesuai dengan ranking hasil perhitungan GVSM terhadap masing-masing dokumen. Dokumen yang memiliki nilai similiralitas 0 adalah dokumen yang tidak sesuai dengan yang dibutuhkan pengguna, sehingga tidak perlu ditampilkan pada pengguna.

3.2 Analisis Sistem IR

Berdasarkan skema proses pencarian pada Gambar 1 diperoleh bahwa proses *tokenizing*, *stop word filtration*, *stemming*, dan *minterm* merupakan proses yang dilakukan berulang-ulang pada setiap dokumen yang ada pada *document collection* dan query masukan pengguna. Perbedaan yang terjadi hanya pada pembacaan masukan query pengguna dan proses pembacaan setiap dokumen yang ada pada *document collection*. Untuk itu, pada penelitian ini, proses yang sama berulang-ulang pada masukan query dan dokumen tersebut, akan dilakukan oleh suatu proses yang berdiri sendiri melalui pengimplementasian *Thread*. Berdasarkan analisis GVSM pada subbab 3.1 maka pada penelitian ini akan mengimplementasikan skema proses pencarian yang mengimplementasikan *multithread* pada

metode GVSM seperti yang dinyatakan pada Gambar 2.



Gambar 2. Skema GVSM *multithread*

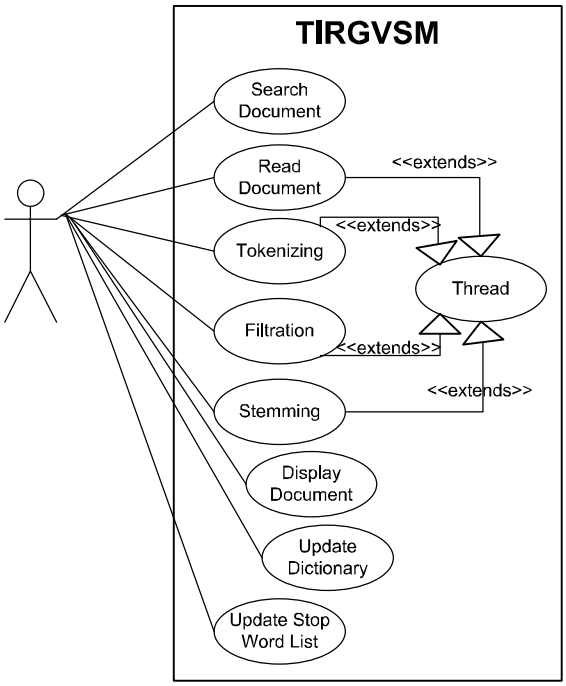
3.3 Analisis Sistem Aplikasi

Untuk memfasilitasi skema pencarian dokumen dengan metode GVSM seperti yang dinyatakan pada Gambar 2, maka sistem akan membutuhkan hal-hal berikut :

1. *User* memasukan *query* untuk mencari dokumen-dokumen yang mempunyai korelasi dengan query dan memilih folder tempat *document collection* yang diinginkan.
2. Sistem akan menciptakan satu *Thread* untuk memproses *query* yang dimasukan oleh *user* dengan memparsing *query* kedalam bentuk *term vector*.
3. Sistem akan menciptakan satu *Thread* terhadap setiap dokumen pada *document collection* dengan melakukan *preprocessing* sehingga dokumen menjadi *term vector*.
4. Hasil *preprocessing* dilanjutkan dengan menghitung pembobotan tiap *term* antar dokumen dengan menggunakan pembobotan *tf-raw*. Selanjutnya menghitung bobot *similarity* antara dokumen dan *query* menggunakan perkalian titik antara vektor dokumen dan vektor *query*.
5. Sistem akan menampilkan dokumen terurut berdasarkan *ranking* yang memiliki nilai similaritas terbesar hasil perhitungan GVSM sedangkan dokumen yang memiliki nilai similaritas *nol* tidak dimasukkan.

3.4 Perancangan Perangkat Lunak

Berdasarkan analisis yang dilakukan pada subbab 3.3, maka untuk memfasilitas kebutuhan pengguna didefinisikan kebutuhan fungsionalitas-fungsionalitas yang meliputi fungsionalitas *search document*, fungsionalitas *read document*, fungsionalitas *tokenizing*, fungsionalitas *filtration*, fungsionalitas *stemming*, fungsionalitas *display document*, fungsionalitas *update* kamus kata dasar, dan fungsionalitas *update stop word list*. Seluruh fungsionalitas aplikasi dinyatakan pada *use case* diagram seperti yang dinyatakan pada Gambar 3.



Gambar 3. Use case diagram aplikasi TIRGVSM

3.5 Implementasi Sistem

Untuk mengimplementasikan perancangan sistem yang dinyatakan pada subbab 3.4 membutuhkan bahasa pemrograman java JDK1.6.7 atau versi yang lebih tinggi. Selain itu juga menggunakan *software* pendukung lainnya NetBeans 7.3 IDE, dan *java class library* yaitu *Apache Poi* yang berfungsi untuk membaca dokumen Microsoft Word, *Xml Beans* berfungsi untuk membaca dokumen berformat *.docx, *dom4J* yang berfungsi untuk membaca dokumen berformat *.doc, serta *PdfBox* yang berfungsi untuk membaca dokumen yang berfomat *.pdf.

3.6 Pengujian Aplikasi

Teknik pengujian yang digunakan adalah teknik pengujian black box testing. Dalam hal ini diambil salah satu butir uji, yaitu pencarian dokumen, seperti yang ditunjukkan pada Tabel 1.

Tabel 1. Pencarian Dokumen

IDENTIFIKASI	
Nomor	TTIRGVSM-01
Nama Butir Uji	Pencarian Dokumen
Tujuan	Mencari dan mengurutkan dokumen sesuai dengan *query* yang dimasukan oleh *user*.
Deskripsi	*User* memasukan *query* pencarian, memilih

	document collection , dan memilih jenis pencarian kemudian menekan tombol *search*.
Kondisi Awal	*User* berada pada menu pencarian dokumen.
Pengujian	

Skenario Uji

1. Memasukan *query* pada *field search*
2. Menentukan *document collection*
3. Memilih pengaturan pencarian pada panel option dengan mencentang pilihan 'Use Thread'
4. Menekan tombol *search*

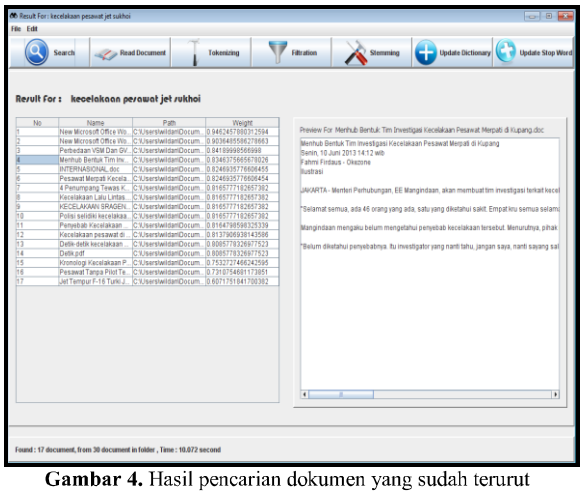
Kriteria Evaluasi Uji

Sistem telah memuat *list* kamus kata dasar bahasa Indonesia dan *list stop word*.

Kasus dan Hasil Uji

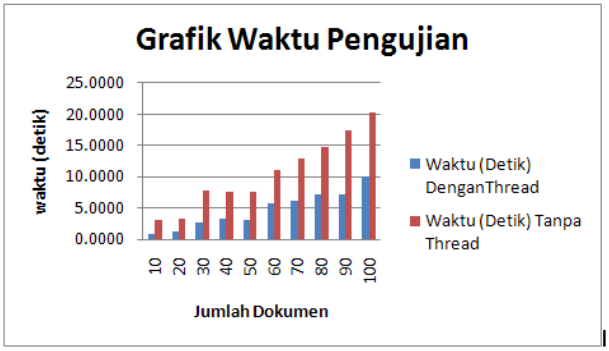
Masukan	Harapan	Pengamatan	Kesimpulan
String Query, String Path Folder *document collection*.	Aplikasi menampilkan dokumen yang memiliki korelasi dengan *query* yang dimasukan oleh pengguna dengan mengurutkan dokumen yang mememiliki bobot yang lebih tinggi hingga bobot yang lebih rendah berdasarkan nilai GVSM yang diperoleh pada setiap dokumen. Dokumen yang nilai similaritas *nol* tidak akan ditampilkan.	Aplikasi menampilkan dokumen yang memiliki korelasi dengan *query* yang dimasukan oleh pengguna dengan mengurutkan dokumen yang mememiliki bobot yang lebih tinggi hingga bobot yang lebih rendah berdasarkan nilai GVSM yang diperoleh pada setiap dokumen. Dokumen yang nilai similaritas *nol* tidak akan ditampilkan.	**[X] Terima** **[ ] Tolak**

Berdasarkan hasil pengujian yang dilakukan pengguna terhadap butir uji Pencarian Dokumen dengan mengikuti skenario yang dinyatakan pada Tabel 1. Diperoleh hasil pengujian seperti yang ditunjukan pada Gambar 4.



Gambar 4. Hasil pencarian dokumen yang sudah terurut

Berdasarkan hasil pengujian yang dilakukan peneliti, aplikasi sudah mampu mencari dan mengurutkan dokumen dari *document collection* dengan baik dengan memperhatikan *precision, recall* dan kebutuhan waktu dengan menggunakan *Thread* dan tanpa menggunakan *Thread* sesuai dengan pemilihan panel option "**Used Thread**". Pada pengujian yang dilakukan, peneliti melakukan pengujian dengan pencarian dokumen dengan menggunakan *Thread* dan tanpa *Thread* pada berbagai *document collection*, kemudian memperhatian *precision, recall*, dan waktu yang dibutuhkan oleh sistem didalam mengurutkan dokumen, seperti yang dinyatakan pada Tabel 2. Adapun grafik waktu yang dibutuhkan oleh aplikasi dengan *multithreading* dan tanpa *multithreading* seperti yang dinyatakan pada Gambar 5.



Gambar 5. Grafik waktu eksekusi

Tabel 2. Hasil Pengujian aplikasi menggunakan Thread dan Tanpa Thread

No	Jumlah Dokumen	Tanpa Thread				Menggunakan Thread				Perbedaan Waktu
		Ditemukan	Precision (%)	Recall (%)	Waktu	Ditemukan	Precision (%)	Recall (%)	Waktu	
1	10	0	100	0	2.995	0	100	0	0.748	2.247
2	20	8	100	40	3.323	8	100	40	1.248	2.075
3	30	9	100	30	7.519	9	100	30	2.746	4.773
4	40	11	100	27.5	7.582	11	100	27.5	3.136	4.446
5	50	15	100	30	7.862	15	100	30	3.183	4.679
6	60	20	100	33.33	11.013	20	100	33.33	5.708	5.305
7	70	22	100	31.43	12.964	22	100	31.43	6.162	6.802
8	80	22	100	27.5	14.687	22	100	27.5	7.114	7.573
9	90	24	100	26.67	17.350	24	100	26.67	7.114	10.236
10	100	24	100	24	20.299	24	100	24	10.047	10.252

4. Kesimpulan

Berdasarkan pengujian yang dilakukan peneliti pada aplikasi *information retrievel* metode GVSM dengan mengimplementasikan *multithreading* dan tanpa *multithreading* diperoleh kesimpulan sebagai berikut :

1. Pada penelitian ini telah berhasil mengembangkan aplikasi IR metode GVSM untuk menemukan kembali dokumen berbahasa Indonesia berformat *.doc, *.docx, dan *.pdf dengan mengimplementasikan *multithreading* maupun tanpa *multithreading*.
2. Aplikasi IR metode GVSM mampu menemukan kembali dokumen yang relevan dan sudah terurut dengan nilai *precision* 100%, baik dengan *multithreading* maupun tanpa *multithreading*.
3. Pengimplementasian *multithreading* pada aplikasi IR metode GVSM dilakukan pada tahap 1 sampai tahap 4 seperti yang dinyatakan pada subbab 3.1, menghasilkan nilai *precision* dan *recall* yang sama.
4. Implementasi *multithreading* pada aplikasi IR metode GVSM mampu meningkatkan kinerja IR, yaitu dapat menghemat waktu proses hingga lebih 50 % seperti yang dinyatakan pada Gambar 5.

Daftar Pustaka:

[1] Goker, A., and Davies, J, (2009), *Information Retrieval : Searching In The 21st Century*, A John Wiley and Sons, Ltd., Publication, United Kingdom.

[2] Ingwersen, I. and Järvelin, K., (2005), *The turn: integration of information seeking and retrieval in context*, Springer.

[3] Kowalski, G., (1997), *Information Retrieval System Theory and Implementation*, Kluwer Academic Publishers, United States of America.

[4] Manning, C., D., et al, (2009). *An Introduction to Information Retrieval*, Cambridge University Press, England.

[5] Nazief, Bobby dan Mirna Adriani, *Confix-Stripping: Approach to Stemming Algorithm for Bahasa Indonesia*, Faculty of Computer Science University of Indonesia.

[6] Oracle, (2012), *Multithreaded Programming Guide*, Oracle and/or its affiliates.

[7] Power, D.M.W, (2011), *Evaluation : From Precision, Recall and F-Measure To Roc, Informedness, Markedness & Correlation*, Journal of Machine Learning Technologies, pp.37-63.

[8] Robertson, S, (2007), *On Document Populations and Measures of IR Effectiveness*, Microsoft Research, Cambridge, UK.

[9] Soboroff, I. and Nicholas, C., (2000), *Collaborative Filtering and The Generalized Vector Space Model*, Athen, Greece, pp.351-353.

[10] Smucker, M.D, and Clarke, C.L.A, (2012), *Time-Based Calibration of Effectiveness Measures*, SIGIR'12, Portland, Oregon, USA.

[11] Tsatsaronis, G. and Panagiotopoulou, V., (2009), *A Generalized Vector Space Model for Text Retrieval Based on Semantic Relatedness*, Proceedings of the EACL 2009 Student Research Workshop, Athen, Greece, pp.70-78.

[12] Wong, S.K.M, Ziarko, W., and Patrick, C.N.W, (1985), *Generalized Vector Space Model In Information Retrieval*, http://www.cs.odu.edu/~jbollen/IR04/ readings/p18-wong.pdf, akses tanggal 27 Desember 2013.

[13] Yates, R.B., and Neto, B.R., (1999*), Modern Information Retrieval*, ACM Press, New York.