IMPLEMENTASI DAN ANALISIS STRUKTUR DATA INDEX SB-TREE PADA TEXT RETRIEVAL SYSTEM

Ardanariswari Skripiyanti[1], Yanuar Firdaus A.w.[2], Warih Maharani[3]

[1]Teknik Informatika, Fakultas Teknik Informatika, Universitas Telkom

Abstrak

Text Retrieval System adalah sistem pencari dokumen teks berdasarkan query masukan pengguna. Sistem pencari dokumen teks terdiri dari dua proses yaitu proses pengindeksan dan pencarian dokumen pada koleksi dokumen di sistem.

Pengindeksan adalah proses mengindeks seluruh term dari seluruh dokumen yang tersimpan pada sistem. Pembangunan indeks bertujuan untuk memudahkan sistem menemukan dokumen yang dicari berdasarkan query. Karena hasil dari pengindeksan tersebut adalah informasi dokumen (umumnya dikodekan berupa id) dimana tiap term yang telah terindeks muncul. Sehingga sistem tidak perlu membaca satu persatu dokumen untuk menemukan informasi yang diinginkan (membutuhkan waktu yang banyak dan proses komputasi yang besar [13]). Informasi tersebut tersimpan pada inverted list.

Pada sebuah sistem pencari teks dengan dokumen yang tersimpan sering berubah (penambahan maupun penghapusan dokumen), dibutuhkan struktur inverted list yang mempercepat proses update inverted list untuk mendukung dynamic indexing.

Salah satu struktur inverted list yang mendukung dynamic indexing adalah SB-tree. SB-tree adalah varian spesial dari struktur B(Bayer)-tree yang memiliki informasi tambahan dan format elemen yang spesial pada leaf node. Update inverted list dilakukan dengan mengunjungi node-node yang mendekati id dokumen yang akan diupdate hingga ditemukan id dokumen tersebut. Sehingga sistem tidak perlu menelusuri seluruh id dokumen secara sekuensial.

Proses pencarian dokumen pada Text Retrieval System dilakukan dengan mencocokkan (merge) inverted list dari setiap term pada query. Pada sistem yang mengimplementasikan struktur SB-tree, algoritma yang digunakan untuk proses merge inverted list, dimulai dengan mencari irisan dari seluruh root node dari setiap term pada query. Sistem akan mengunjungi node di bawahnya jika node tersebut beririsan dengan node dari seluruh term lainnya, demikian seterusnya hingga ke leaf node. Dengan algoritma tersebut kinerja sistem dalam menemukan dokumen menjadi efektif dan efisien.

Kata Kunci : Text Retrieval, Text Retrieval System, indexing dan SB-tree.

Abstract

Text Retrieval System is used to search text document which relevan to query. Text Retrieval System consists of indexing and searching text document stored in system.

Indexing is a process of storing indices of the term within the document stored in system. It is need a data structure to store the index. To improved performance, this data structure is able to support dynamic indexing, which is especially important for environments where documents are changed frequently.

One of the data structure is called SB-tree. SB-tree is a special variant of B(Bayer)-tree with additional information and special element format that is stored in leaf node.

To update inverted list, system use the bounding boxes from the nodes to ensure that "nearby" elements are placed in the same leaf node (in particular, a new element will go into the leaf node that requires the least enlargement in its bounding box).

Keywords : Text Retrieval, Text Retrieval System, indexing and SB-tree

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Information Retrieval System adalah sistem pencari yang melakukan proses pencarian informasi secara otomatis berdasarkan isi maupun konteks tertentu sesuai dengan *query* inputan pengguna.

Proses yang dilakukan sistem pencari informasi yang baik, pada umumnya terdiri dari beberapa langkah berikut [2] :

1. *Query pre-processing*, salah satunya adalah proses normalisasi bahasa pada *term*.
2. Proses pembuatan indeks pada pembangunan sebuah sistem pencari.
3. *Post-processing* pada dokumen kandidat yang akan dipilih sebagai dokumen yang dianggap relevan terhadap *query*. Salah satunya dengan merangking dokumen yang relevan terhadap *query*.
4. Perbaikan *query* berdasarkan pada *feedback* dari *user* dan re-evaluasi terhadap *query*.

Langkah tersebut sangat berperan penting terhadap kualitas hasil yang ditampilkan oleh sistem kepada pengguna. Namun secara garis besar sistem pencari informasi terdiri dari dua tahapan yaitu proses *indexing* dan *searching*.

Proses pengindeksan adalah proses mengindeks *term* dari seluruh dokumen yang tersimpan pada sistem. Hasil pengindeksan tersebut disimpan pada basis data indeks dalam bentuk *inverted list* untuk setiap *term*. Dengan kata lain, *inverted list* menyimpan sejumlah *posting(id* dokumen*)* dimana setiap *term* yang telah terindeks muncul.

Indeks dibuat untuk memudahkan pencarian dokumen. Sehingga sistem tidak perlu membaca satu persatu dokumen untuk menemukan informasi yang diinginkan.

Pada sebuah sistem pencari dengan dokumen yang tersimpan sering berubah (penambahan maupun penghapusan dokumen), maka struktur *inverted list* yang digunakan sebaiknya mendukung *dynamic indexing* (indeks yang bersifat dinamis; adanya penambahan maupun penghapusan *id* dokumen pada *inverted list*).

Pada Tugas Akhir ini dipelajari sebuah struktur data untuk penyimpanan *inverted list* yang memungkinkan *dynamic indexing* yaitu struktur SB-tree.

1.2 Perumusan Masalah

Berdasarkan uraian diatas, maka permasalahan yang muncul dan yang menjadi objek penelitian pada Tugas Akhir ini ialah:

1. Bagaimana mengimplementasikan sebuah struktur penyimpanan *inverted list* yang mendukung *dynamic indexing* untuk sistem pencari dokumen teks dengan dokumen yang sering berubah.
2. Dari struktur yang digunakan tersebut, bagaimana sistem melakukan proses *merge inverted list* untuk menemukan dokumen yang relevan terhadap *query*.

Batasan masalah agar tidak meluasnya materi pembahasan dalam tugas akhir ini ialah:

2. Analisis dilakukan pada *document collection* (tersimpan pada *disk*) yang bersifat statis.
3. *Document collection* pada sistem yaitu berupa *text document*.

1.3 Tujuan

Secara umum tujuan penulisan yang ingin dicapai dalam Tugas Akhir ini ialah:

1. Menerapkan struktur data SB-tree untuk penyimpanan *inverted list* pada sistem pencari dokumen teks yang mendukung dilakukan *dynamic indexing*.
    - menganalisis ukuran basis data yang dibutuhkan untuk menyimpan *inverted list* dengan struktur SB-tree.
    - menganalisis waktu yang dibutuhkan untuk proses update *inverted list* pada struktur SB-tree.

2. Menerapkan algoritma khusus yang memanfaatkan struktur SB-tree untuk melakukan proses *merge inverted list* secara efektif dan efisien.
    - menganalisis waktu yang dibutuhkan untuk *merge inverted list* pada struktur SB-tree.
    - menganalisis pengaruh frekuensi *term* pada *query* terhadap nilai Performance Gain Estimation.
    - menganalisis pengaruh ukuran *block size* sistem terhadap kinerja sistem dalam proses *merge inverted list.*

1.4 Metodologi Penyelesaian Masalah

Metodologi yang digunakan untuk menyelesaikan masalah dalam Tugas Akhir ini ialah:

1. Studi literatur.
   Melakukan diskusi dengan dosen pengajar dan mencari permasalahan yang terjadi pada pemrosesan *indexing* pada sistem pencari teks. Kemudian mencari algoritma sebagai solusinya dan mengumpulkan informasi yang terkait dengan proses *indexing* dan memahami konsep algoritma yang digunakan dari struktur data SB-tree pada proses *indexing* melalui literatur berupa makalah, buku, maupun jurnal berupa dokumen elektronik maupun fisik.

2. Pencarian dan pengumpulan data.
   Data yang akan digunakan berupa dokumen elektronik berbahasa Inggris yang diperoleh secara bebas dari internet.
3. Analisis kebutuhan dan perancangan aplikasi yang akan dibangun.
   Menganalisa kebutuhan perangkat lunak dan merancang perangkat lunak untuk implementasi struktur data SB-tree pada proses *indexing*.
4. Implementasi dan pengujian
   Mengimplementasikan hasil analisis dan perancangan perangkat lunak dengan menggunakan teknik berorientasi objek serta melakukan pengujian dan pengukuran performansi dari sistem.
5. Analisis hasil pengujian dan membuat kesimpulan
   Perangkat lunak yang dihasilkan dievaluasi berdasarkan data yang diperoleh dari hasil pengujian.
6. Penyusunan laporan tugas akhir.
   Pembuatan laporan tugas akhir yang mendokumentasikan tahap-tahap kegiatan dan hasil dalam tugas akhir ini.

1.5 Sistematika Penulisan

Sistematika penulisan Tugas Akhir ini terdiri dari 5 Bab, yaitu:

BAB I Pendahuluan

Bab ini membahas kerangka penelitian dalam tugas akhir, meliputi latar belakang, perumusan masalah, batasan masalah, tujuan perancangan dan metodologi yang digunakan dalam perancangan sistem.

BAB II Landasan Teori

Bab ini menjelaskan seluruh teori yang menjadi landasan konseptual dan mendukung penyelesaian tugas akhir ini.

BAB III Perancangan Sistem

Bab ini membahas mengenai pengumpulan data analisis dan perancangan sistem yang terdiri dari perancangan alur kerja sistem, perancangan basis data, perancangan *interface* dan perancangan modul pada perangkat lunak.

BAB IV Implementasi, Pengujian dan Analisis Sistem

Bab ini membahas implementasi detail sistem, pengujian terhadap sistem dan menganalisis hasil pengujian tersebut untuk mengetahui kesimpulan dari metode yang diimplementasikan.

BAB V Kesimpulan dan Saran

Berisi tentang kesimpulan dan saran yang dapat diambil dari keseluruhan sistem yang telah dibuat.

BAB 5

KESIMPULAN DAN SARAN

Pada bab ini akan diuraikan hal yang dapat disimpulkan dari pelaksanaan Tugas Akhir ini. Selain itu diuraikan pula beberapa saran yang dapat digunakan dalam pengembangan Tugas Akhir di masa mendatang.

5.1 Kesimpulan

Berdasarkan hasil analisis dan pengujian perangkat lunak yang dilakukan dalam tugas akhir ini dapat diambil beberapa kesimpulan, yaitu:

1. Dari implementasi *inverted list* dengan struktur SB-tree dapat disimpulkan bahwa ukuran basisdata dipengaruhi oleh jumlah *record* yang tersimpan. Jumlah *record* yang tersimpan dipengaruhi oleh jumlah *term* yang terindeks dan panjang *inverted list* setiap *term* tersebut.
2. Struktur *inverted list* SB-tree mendukung *dynamic indexing* pada Text Retrieval System dengan dokumen yang terindex sering berubah. Update *inverted list* dilakukan dengan mengunjungi *node-node* yang berdekatan nilai elemennya terhadap nilai elemen yang dicari, dengan menggunakan informasi *bounding box* yang ada di setiap *non-leaf node.* Sehingga sistem dengan cepat menemukan elemen yang dicari pada *leaf node.*
3. Waktu yang dibutuhkan untuk proses *merge inverted list* dengan algoritma khusus yang memanfaatkan struktur SB-tree dipengaruhi oleh banyaknya *term* pada *query.* Dan panajang *inverted list* dari setiap *term* tersebut.
4. Faktor yang dapat memperkecil kinerja sistem menemukan dokumen dengan implementasi algoritma *merge inverted list* khusus yang memanfaatkan struktur SB-tree adalah ketepatan sistem dalam menemukan calon dokumen hasil dari *stack* pada proses *merge inverted list.*
   Nilai PGE berhubungan dengan kemampuan sistem dalam menemukan dokumen berdasarkan *query*. Semakin besar ukuran *block size*, maka semakin baik performansi sistem, ditandai dengan nilai PGE yang semakin kecil.

5.2 Saran

Untuk pengembangan Tugas Akhir di masa mendatang, penulis menyarankan hal-hal sebagai berikut:

1. Ukuran *inverted list* yang semakin besar, akan berpengaruh pada ukuran ruang penyimpan yang digunakan. Salah satu solusinya adalah dengan

melakukan kompresi terhadap *inverted list.* Alangkah baiknya jika kompresi dilakukan terhadap nilai *run-length.* Karena hal ini akan semakin memperkecil ukuran *inverted list.*

2. Lebih baik jika *document collection* yang digunakan bersifat dinamis.

DAFTAR PUSTAKA

[1]	Alistair Moffat and Justin Zobel. Self-Indexing Inverted Files for Fast Text Retrieval. In Australian Database Conf. and IEEE Conference on Data Engineering., 1994. Berisi tentang pengurangan response time pada query processing/query evaluation.
[2]	An Indexing Algorithm for Text Retrieval, http://meta.math.spbu.ru/publication, didownload pada tanggal 13 Maret 2009. Berisi tentang struktur index SB-tree serta algoritma yang digunakan dalam proses penggabungan inverted list pada Text Retrieval System.
[3]	D. Comer, "The Ubiquitous B-Tree," ACM Computing Surveys 11(2): 121-137(1979). Berisi tentang B-tree.
[4]	Dynamic indexing information retrieval or filtering system, http://www.freepatentsonline.com, didownload pada tanggal 18 Agustus 2009. Berisi tentang pengertian dynamic indexing pada Information Retrieval.
[5]	E.W. Brown, J.P. Callan, W.B. Croft, and J.E.B. Moss. Supporting Full - text information retrieval with a persistent object store,. In Proc. Intnl.Conf. on EDBT., 1994. Berisi tentang pengindeksan inverted file.
[6]	Gerald Huck, Frank Moser, and Erich J. Neuhold. Integration and handling of hypermedia information as a challenge for multimedia and federated database systems. In *Proc. of the Second Intnl. Workshop on Advances in Databases and Information Systems - ADBIS'95*, pages 183–194, Moscow, June 27–30 1995. Phasis. Berisi tentang penyimpanan record data berukuran besar pada database.
[7]	Information Retrieval, http://www.ittelkom.ac.id/staf/yanuar, didownload pada tanggal 2 September 2008. Berisi tentang konsep dasar Information Retrieval.
[8]	Justin Zobel, Alistair Moffat, and Ron Sacks-Davis. Efficient indexing technique for full-text database systems. In Proc. 18th Intnl.Conf. on VLDB. Vancouver, British Columbia, Canada, 1992., pages 352–362, 1992. Berisi penjelasan inverted file pada proses pengindeksan dan salah satu cara memperkecil ukuran inverted file.

[9]	Justin Zobel, Alistair Moffat, and Ron Sacks-Davis. Searching large lexicons for partially specified terms using compressed inverted files. In *Proc. 19th Intnl.Conf. on VLDB. Dublin, Ireland, 1993.*, pages 290–301, 1992. Berisi tentang cara menggunakan inverted file index hasil kompresi untuk mencari kata dalam dictionary pada sistem pencari.
[10]	Show Table Status Sintax, http://www.mysql.com, didownload pada tanggal 26 Agustus 2009. Berisi tentang sintax sql untuk mengetahui informasi non-temporary table pada basisdata MySQL.
[11]	The R*-tree: an efficient and robust access method for points and rectangles, http://citeseer.ist.psu.edu/36844.html, didownload pada tanggal 13 Maret 2009. Berisi tentang R-tree dan parameter penting yang mempengaruhi performansi Information Retrieval System.
[12]	http://en.wikipedia.org/wiki/Information_retrieval.htm didownload pada 1 Oktober 2009. Berisi tentang pencarian query pada information retrieval.
[13]	http://en.wikipedia.org/wiki/Index_(search_engine).htm didownload pada 1 Oktober 2009. Berisi tentang proses indexing pada Information Retrieval.
[14]	http://en.wikipedia.org/wiki/Inverted_index.htm didownload pada 1 Oktober 2009. Berisi tentang inverted index pada Information Retrieval.
[15]	http://www.arwankhoiruddin.co.cc/files/Tree.doc didownload pada 5 Oktober 2009. Berisi tentang struktur tree.
[16]	http://en.wikipedia.org/wiki/R-tree.htm didownload pada 1 Oktober 2009. Berisi tentang struktur R-tree.
[17]	http://en.wikipedia.org/wiki/B-tree.htm didownload pada 1 Oktober 2009. Berisi tentang struktur B-tree.
[18]	http://en.wikipedia.org/wiki/accesstime.htm didownload pada 16 Oktober 2009. Berisi tentang waktu pengaksesan disk.

[19]	http://www.wikipedia.org/wiki/01bool.pdf didownload pada 3 November 2009. Berisi tentang Boolean Retrieval.