RANCANG BANGUN APLIKASI PENCARIAN DOKUMEN BERBASIS WEB MENGGUNAKAN METODE SUFFIX CACTUS CLUSTERING

F.X. Arunanto dan Agus Widodo

Jurusan Teknik Informatika, Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember (ITS) - Surabaya
Kampus ITS, Jl. Raya ITS, Sukolilo – Surabaya 60111
Tel. + 62 31 5939214, Fax + 62 31 5939363
Email : anto @its-sby.edu

ABSTRAK

Salah satu bagian dari temu kembali (retrieval) informasi adalah menyajikan data hasil pencarian dengan cara mengelompokkan (clustering) seluruh dokumen hasil pencarian sesuai dengan kemiripan antar dokumen. Agar tingkat kemiripan dokumen tinggi maka dokumen harus dibaca dengan memperhatikan struktur frasa-frasa penyusun dokumen (semantis).

Pembacaan secara semantis inilah yang kemudian membutuhkan representasi isi dokumen secara tepat. Hal ini dipecahkan dengan preprocessing dokumen hasil pencarian dalam bentuk struktur data suffix cactus. Implementasi struktur data suffix cactus memungkinkan pencarian string relatif lebih cepat dibandingkan query biasa. Suffix cactus bisa dikonstruksi dari suffix tree ataupun dari suffix array. Kinerja suffix cactus berada di antara keduanya.

Pada makalah ini, suffix cactus dikonstruksi berdasarkan pada informasi yang didapat dari suffix tree, sedangkan clustering diimplementasikan pada judul dokumen yang dianggap representasi dokumen berbahasa Indonesia. Hasil pencarian dokumen berupa grup-grup dokumen dan ditampilkan frasa-frasa baru yang memiliki kedekatan dengan keyword yang diinputkan. Pada makalah ini juga dilakukan pengukuran waktu eksekusi dan ukuran suffix cactus terhadap jumlah dokumen (hits).

Kata kunci*: Information Retrieval, clustering, suffix tree, suffix cactus*.

1. PENDAHULUAN

Munculnya temu kembali informasi (i*nformation retrieval*) sebagai cabang ilmu baru dalam Teknologi Informasi, memperkenalkan banyak hal baru terkait dengan pencarian informasi. Pencarian informasi berbasis *query (query based retrieval)* yang digunakan secara tradisional sangat berguna untuk pencarian terarah tetapi tidak begitu efisien untuk *generic query*[4]. *Query Based Retrieval* berguna ketika kita mengetahui benar informasi yang dicari. Cara ini tidak begitu efisien ketika kita membutuhkan informasi yang spesifik/khusus dalam kategori yang besar[11].

Sebuah *query* pada suatu sistem retrieval dokumen, umumnya memberikan hasil berupa cuplikan dokumen-dokumen yang disusun berdasarkan tingkat kecocokan(*matching*) dengan *query* yang diinputkan[2]. Tidak jarang pencarian menghasilkan urutan dokumen-dokumen yang terlalu panjang dan menyulitkan jika dicek satu persatu. Dokumen yang ditampilkan sebagai hasil pencarian banyak yang memiliki tingkat kecocokan yang rendah. Akan lebih efisien jika dokumen-dokumen hasil pencarian dikelompokkan menjadi grup-grup atau cluster sehingga pencarian akan lebih mudah ditelusuri.

Ada beberapa metode terkait dengan clustering pada pencarian dokumen. Diantara metode-metode tersebut adalah metode Suffix Tree *Clustering* (STC). Ide dasar metode ini adalah dengan mengelompokkan dokumen ke dalam bentuk grup-grup atau cluster berdasarkan frasa-frasa yang dipakai bersama (*shared phrases*) dalam dokumen-dokumen tersebut[2]. Cluster-cluster inilah yang akan ditampilkan ke pengguna. Struktur data Suffix Tree mempergunakan cukup banyak pointer. Permasalahan mungkin dapat timbul jika dokumen sangat besar sehingga *tree* tidak dapat ditampung dalam memory yang menyebabkan proses *paging*[3]. Hal ini dapat diatasi dengan mempergunakan suatu struktur data lain yang hasilnya ekivalen dengan Suffix Tree tetapi dengan waktu konstruksi yang lebih besar, yakni dikenal dengan metode Suffix Array. Suatu bentuk hibrid dari kedua struktur data ini disebut Suffix Cactus[1].

Konstruksi Suffix Cactus bisa dilakukan melalui Suffix Tree ataupun Suffix Array. Dalam Tugas Akhir ini dilakukan konstruksi Suffix Cactus dari Suffix Tree.

2. SISTEM TEMU KEMBALI INFORMASI

Sistem temu kembali informasi dapat didefinisikan sebagai suatu sistem bantu yang bertujuan menyajikan informasi dalam hal eksistensi obyek yang dianggap relevan dan meminimalkan informasi yang tidak relevan terhadap suatu *query* yang merupakan hasil pemrosesan tertentu dari sekumpulan informasi. *Query* dalam konteks ini adalah suatu formulasi yang dikonstruksi mengikuti kaidah-kaidah tertentu dan merefleksikan kebutuhan akan suatu informasi.

Keefektifan dari temu kembali informasi yang diinginkan tergantung pada dua hal dasar yaitu perilaku pengguna dan *logical view* dari sistem temu kembali.

Pengguna suatu sistem temu kembali pada umumnya harus menerjemahkan kebutuhan informasinya ke dalam suatu *query* dengan bahasa tertentu yang disediakan oleh sistem. Pada sistem temu kembali hal ini berarti menspesifikasikan sekumpulan kata-kata yang mewakili informasi yang dibutuhkan. Terdapat dua proses yang dominan dalam tingkah laku pengguna dalam mencari informasi yaitu temu kembali (*retrieval*) dan *browsing*. Untuk temu kembali dokumen tertentu yang diinginkan oleh pengguna. pengguna menggunakan *query* untuk mendapatkan informasi yang diinginkan.

Faktor kedua adalah *logical view* tentang representasi dokumen dalam bentuk koleksi keseluruhan kata-kata yang dikandungnya. *Full text* merupakan *logical view* paling lengkap dengan melakukan eliminasi *stoplist* (kata-kata yang tidak perlu seperti kata sandang, kata sambung, kata ganti dan lain-lain) dan *stemming* (pengubahan bentuk jadian ke bentuk dasar).

3. CLUSTERING

Clustering adalah istilah dalam bahasa Inggris yang berarti usaha atau aktifitas untuk membentuk cluster. Cluster sendiri adalah suatu grup tertutup atau sekumpulan obyek yang mirip (*similar*). *Clustering* terutama digunakan untuk klasifikasi data. *Clustering* telah terbukti menjadi unsur penting dalam pengolahan data terutama data-data mentah yang diinginkan untuk menjadi suatu bentuk pola-pola kumpulan data.

Clustering hasil pencarian dapat membantu penguna dalam tiga hal, yakni pertama, memudahkan pencarian informasi yang dibutuhkan, kedua, membantu pengguna sehingga lebih cepat menyadari bahwa *query* yang diformulasikan kurang tepat (misalnya, bersifat terlalu umum). Ketiga, mengurangi fraksi atau tingkat kuantitas tertentu dimana pengguna 'menyerah' sebelum menemukan informasi yang diinginkan [2].

Kebutuhan (*requirement*) utama clustering dokumen hasil mesin pencarian adalah sebagai berikut : [2]

[1]. Relevansi : Sistem mampu menghasilkan cluster yang berisi dokumen-dokumen yang relevan dengan tidak mengikutkan dokumen-dokumen yang tidak relevan dengan *query* pengguna.

[2]. Deskripsi : Cluster yang dihasilkan memiliki deskripsi yang akurat mengenai dokumen-dokumen di dalamnya. Deskripsi ini digunakan oleh pengguna untuk memilih cluster yang paling dekat dengan kebutuhannya.

[3]. Kecepatan : Pada sistem temu kembali yang bersifat *online*, kecepatan respon merupakan hal yang signifikan dalam menentukan kinerja sistem. Dengan diterapkannya algoritma clustering, waktu pemrosesan tidak boleh bertambah secara signifikan. Untuk itu, selain dibutuhkan algoritma yang cepat dan *scalable*, juga dibutuhkan toleransi cuplikan, yakni bahwa cluster dapat dibuat berdasarkan cuplikan dokumen dimana cluster diterapkan pada keseluruhan dokumen.

3.1. Suffix Cactus Clustering

Perlu dijelaskan tentang keterkaitan antara Suffix Tree dan Suffix Cactus karena keduanya memiliki banyak kesamaan. Sedang mengenai Suffix Array hanya sebagai tambahan yang bisa menjelaskan bahwa Suffix Cactus adalah bentuk *hybrid* dari Suffix Tree dan Suffix Array.

Suffix Tree merupakan salah satu dari struktur data yang paling penting pada *stringology*. Suffix Tree adalah struktur semacam index yang dibentuk dari sebuah string yang memungkinkan banyak *query* yang cepat terhadap string. Apa yang membuat Suffix Tree menarik adalah bahwa ukuran dan waktu konstruksinya yang linear terhadap panjang teks[1]. Suffix Tree banyak diterapkan pada banyak ragam aplikasi.

Aplikasi tersebut merupakan implementasi Suffix Tree sebagai indeks teks statis berukuran besar yang memungkinkan pencarian yang cepat. Tipe pencarian dasarnya adalah pencocokan (*matching*) string seperti pencarian untuk kejadian dari sebuah pola string pada teks. Contoh-contoh teks berukuran sangat besar yang memerlukan pencarian cepat adalah kamus elektronik dan database rangkaian biologis.

Agar berjalan efisien, keseluruhan Suffix Tree harus diletakkan pada memory utama. Jadi bahasan penting yang kemudian muncul adalah kebutuhan akan ruang memory. Ukuran tepat Suffix Tree tergantung pada implementasi dan tipe teks. Pada implementasi dengan teks bahasa Inggris, ukuran

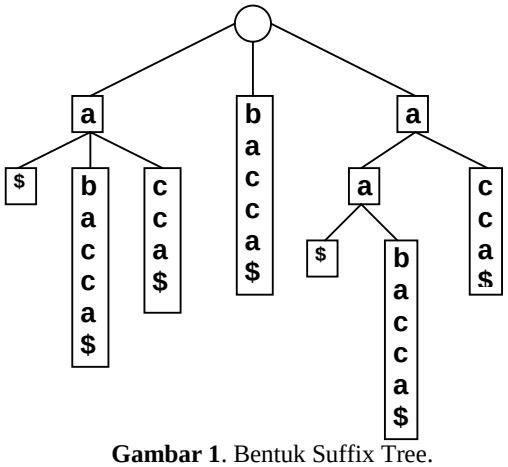
dari sebuah Suffix Tree adalah 15 bytes per simbol teks.

Suffix Array adalah sebuah sebuah struktur data, mirip dengan Suffix Tree, yang mengijinkan pencarian cepat pada sebuah teks. Ukuran dari sebuah implementasi Suffix Array yang efisien hanya membutuhkan 6 bytes per simbol teks. Pada pencocokan string, kinerja Suffix Array dapat diperbandingkan dengan Suffix Tree. Akan tetapi pada tipe pencarian yang lain seperti pada pencocokan *regular expression*, Suffix Array membutuhkan waktu lebih banyak(lebih lambat).

Satu lagi struktur data yang akan dibahas adalah Suffix Cactus yang mirip dengan Suffix Tree. Ukuran dari sebuah Suffix Cactus sebesar 10 bytes per simbol teks. Jika kita bandingkan dengan ukuran Suffix Tree dan Suffix Array, Suffix Cactus berada antara keduanya. Kinerja pada banyak aplikasi sama dan tetap pada *regular expressions matching*.

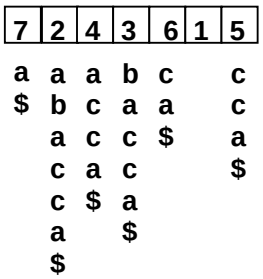
Suffix Cactus memberikan sudut pandang baru yang menarik pada keluarga struktur data suffix. Struktur dari Suffix Cactus memiliki kesamaan dengan struktur pada Suffix Tree maupun pada Suffix Array. Suffix Cactus dapat dideskripsikan sebagai versi *compact*(rapat) Suffix Tree atau juga sebagai sebuah Suffix Array yang diperbesar dengan adanya beberapa informasi tambahan. Sehingga Suffix Cactus dapat disebut sebagai persilangan atau hibrid antara Suffix Tree dan Suffix Array.

Agar lebih jelas, berikut akan digambarkan bentuk Suffix Tree, Suffix Cactus dan Suffix Array untuk string **cabacca$**

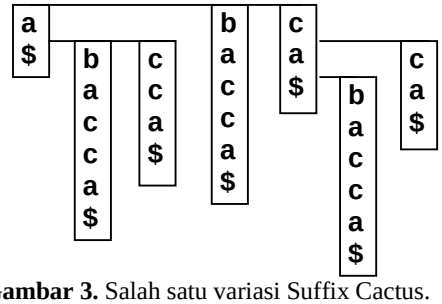


Gambar 1. Bentuk Suffix Tree.

Nama cactus diambil dari cara *branch-branch* memulai di tengah-tengah *branch* yang lain. Alternatif definisi apapun dari struktur tree yang digunakan, jenis *branch*ing memerlukan implementasi yang berbeda dari *branch*ing tree tradisional. Implementasi akan berpengaruh pada kebutuhan ruang secara tepat pada Suffix Cactus dan kompleksitas waktu pada permasalahan-permasalahan matching yang berbeda.



Gambar 2. Bentuk Suffix Array.



Gambar 3. Salah satu variasi Suffix Cactus.

Terdapat tiga fase untuk clustering dengan menggunakan Suffix cactus. ***Fase pertama*** adalah pembentukan Suffix Tree dari kumpulan string dokumen yang cocok dengan mode pencarian. ***Fase kedua*** adalah identifikasi cluster frasa dengan pembacaan Suffix Tree untuk mendapatkan nilai DEPTH dan SUFFIX. Representasi Suffix Cactus didapatkan setelah proses konstruksi nilai SIBLING yang diperoleh dari DEPTH.. Kemudian dideteksi *node parent* sebuah *branch*. Kita bisa mendapatkan cluster frasa dengan mencari LCP (*Longest Common Phrase*) sebagai prefiks maksimal antara *child* dan *parent*-nya. .

Setelah cluster frasa maksimal teridentifikasi maka kita perlu memberi skor untuk masing-masing cluster frasa yang ditemukan. Pemberian skor berdasarkan suatu fungsi dengan parameter jumlah dokumen yang terkandung di dalamnya dan frasanya. Skor suatu cluster frasa dipergunakan untuk menemukan 'kelayakan' suatu cluster frasa. Hal ini sangat diperlukan dalam proses clustering. Fungsi skor *s(m)* suatu cluster frasa *m* dengan frasa m_p diberikan sebagai berikut :

$$S(m) = |m| \, . \, f(|m_p|)$$

dimana $|m|$ adalah jumlah dokumen dalam cluster frasa *m*. $|m_p|$ adalah jumlah kata dalam m_p. Fungsi f bersifat linear untuk frasa yang terdiri dari 2 hingga

6 kata dan bernilai konstan untuk frasa yang lebih panjang. Akhirnya dipilih *k* cluster frasa dengan skor paling tinggi. Variabel *default k* ditentukan sebesar 10% atau 0.1 dari seluruh jumlah cluster yang ditemukan.

Fase ketiga adalah penggabungan cluster-cluster frasa. Hal ini diperlukan karena cluster-cluster frasa yang diidentifikasi bersifat *overlap* bahkan dapat identik. Cluster-cluster frasa yang digabung adalah cluster-cluster dengan tingkat *overlapping* yang tinggi.

Dilakukan perhitungan kemiripan (*similarity measure*) biner antar cluster frasa berdasarkan dokumen-dokumen yang *overlap* dalam cluster tersebut.

$sim(m_i,m_j) = 1$,
jika $|m_i \cap m_j| / |m_i| > \alpha$
dan $|m_i \cap m_j| / |m_j| > \alpha$
$sim(m_i,m_j) = 0$ jika sebaliknya.

Dari sini dapat dibentuk *merged clusters* yang berisi gabungan (*union*) dokumen-dokumen dari semua cluster frasa-nya.

4. PERANCANGAN DAN IMPLEMENTASI

Dokumen harus dibaca secara semantis (memperhatikan frasa-frasa penyusunnya) agar 'kandungan' dokumen lebih bisa dirasakan oleh user. Permasalahan yang kemudian muncul adalah

- Perlunya representasi tree yang benar dan efisien untuk proses penelusuran frasa yang berdekatan dan dipakai bersama oleh beberapa dokumen.
- Struktur yang berbeda dan khas antar bahasa yang menyebabkan perlu dilakukan perbaikan kualitas dokumen.

Untuk memperbaiki kualitas dokumen, dilakukan beberapa langkah. Yang pertama, dihilangkan seluruh kata yang termasuk karakter yang tidak perlu dan *stoplist* pada dokumen. *Stoplist* disini adalah kata yang sering muncul dan tidak bisa dijadikan pembeda dokumen. Langkah kedua dilakukan stemming untuk menjadikan kata jadian menjadi kata dasar.

Dalam morfologi kata Bahasa Indonesia dikenal adanya tiga imbuhan yaitu walan(prefiks), sisipan, dan akhiran(sufiks). Untuk penanganan dokumen yang mengandung kata jadian pada tugas akhir ini hanya akan menghilangkan awalan dan akhiran.

Metode ini didahului dengan pembacaan tiap kata dari file sampel. Sehingga input dari algoritma ini adalah sebuah kata yang kemudian dilakukan pemeriksaan semua kemungkinan bentuk kata. Setiap kata dalam Bahasa Indonesia umumnya maksimal memiliki dua awalan (*prefiks*) dan tiga akhiran (*sufiks*).
Sehingga bentuknya menjadi :

P1 + P2 + KD + S3 + S2 + S1

Aplikasi mengambil contoh kasus pencarian karya-karya tugas akhir atau tesis mahasiswa T.Informatika ITS yang tersimpan dalam suatu database Ruang Baca Teknik Informatika. Maksudnya adalah bahwa aplikasi ini tidak dirancang untuk melakukan pencarian dalam internet, melainkan langsung melakukan pencarian ke dalam database. Walaupun berada pada lingkungan tertutup, arsitektur *client-server* yang dipergunakan dan antarmuka web memungkinkan proses pencarian dilakukan dari berbagai lokasi.

Skema basis data yang mengikuti representasi data sebenarnya yang ada pada tabel 'tesis' pada database RBTC Data yang ada meliputi judul, penulis, nama penulis, tipe, NRP penulis, tahun penulisan,pembimbing I, pembimbing II, dan abstrak.

Data yang akan dijadikan sebagai area pencarian adalah judul tesis. Tentunya representasi frasa yang akan ada ditemukan juga frasa-frasa yang bisa ditemukan pada judul tesis tersebut.

Aplikasi diimplementasikan dengan mempergunakan Oracle 8i sebagai DBMS server, Apache sebagai web server, *indexing* dilakukan dengan menggunakan *inter*Media Text dan pada sisi klien mempergunakan *browser* Internet Explorer.
Pada tahap implementasi dilakukan pembuatan dan populasi tabel-tabel yang telah dirancang. Setelah itu, data dalam tabel ditransformasi ke dalam bentuk indeks dengan mengikutkan aturan-aturan tertentu, misalnya kata-kata apa saja yang tidak akan diikutkan dalam proses indexing (*stopword*). Implementasi antarmuka dengan membuat template *.htm untuk file *.php yang bersesuaian (dengan nama yang sama).

5. UJI COBA

Uji coba dilakukan pada database berisi 566 dokumen. Data berupa tabel dengan nama 'tesis' yang dipakai sebagai penyimpanan data tugas akhir dan tesis mahasiswa Teknik Informatika ITS. Dilakukan input berupa *keyword* yang sama yaitu dengan mempergunakan *keyword* 'asosia'.

Dilakukan dua bentuk uji coba. Uji coba pertama dipergunakan untuk mengetahui pembentukan Cluster Frasa dan proses penggabungannya. Sedang uji coba kedua bertujuan untuk mengetahui hubungan antara *hits* (jumlah dokumen hasil pencarian) dengan waktu eksekusi

serta hubungan antara *hits* dengan *space* (ukuran Suffix cactus).

Input pada uji coba pertama tidak bersifat *case sensitive*, artinya keyword 'ASOsia' dan 'asosia' akan memberikan hasil yang sama. Dengan keyword 'asosia' maka aplikasi akan mencari dokumen-dokumen yang memiliki substring 'asosia'.

Cluster frasa haruslah dipilih dengan 'kualitas' yang baik. Terdapat 4 cluster frasa yang bisa digunakan untuk penelusuran selanjutnya. Empat frasa tersebut adalah :

1. 'asosiasi' yang ada pada dokumen dengan ID : 365, 2231, 2253, 2225, 394
2. 'kaidah asosiasi' yang ada pada dokumen dengan ID : 2231, 2225, 2253
3. 'pencarian kaidah asosiasi' yang ada pada dokumen dengan ID : 2231, 2225
4. 'pencarian pola-pola asosiasi' yang ada pada dokumen dengan ID : 1960, 365

Dari fase cluster yang ditemukan, semuanya akan dipakai untuk menentukan cluster-cluster. Penentuan ini berdasarkan perhitungan kemiripan (*similarity measure*) yang diperoleh dari perhitungan pada dokumen-dokumen yang terdapat pada tiap-tiap cluster frasa.

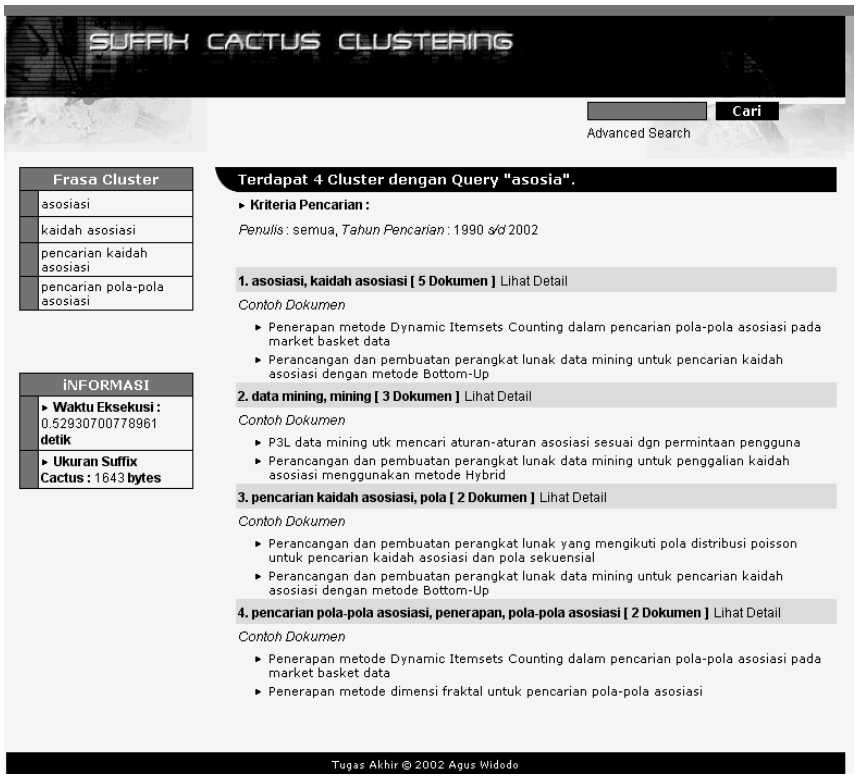
Diperoleh sejumlah 4 cluster yaitu :

- Cluster ke-1
  Cluster frasa : 'asosiasi', 'kaidah asosiasi'
  Jumlah dokumen anggota : 5
  ID dokumen : 365, 2231, 2253, 2225, 394
- Cluster ke-2
  Cluster frasa : 'data mining', 'mining'
  Jumlah dokumen anggota : 3
  ID dokumen : 2231, 2253, 394
- Cluster ke-3
  Cluster frasa : 'pencarian kaidah asosiasi', 'pola'
  Jumlah dokumen anggota : 2
  ID dokumen : 2231, 2253
- Cluster ke-4
  Cluster frasa : 'pencarian pola-pola asosiasi', 'penerapan', 'pola-pola asosiasi'
  Jumlah dokumen anggota : 2
  ID dokumen : 365, 1960

Yang belum dilakukan sampai tahap ini adalah pemberian skor cluster. Cluster-cluster yang dibentuk ditampilkan berdasarkan urutan dalam tree bukan berdasarkan skor.

Informasi anggota cluster untuk cluster ke-1 didapatkan dokumen-dokumen dengan judul:

1. 'Penerapan metode Dynamic Itemsets Counting dalam pencarian pola-pola **asosiasi** pada market basket data'.

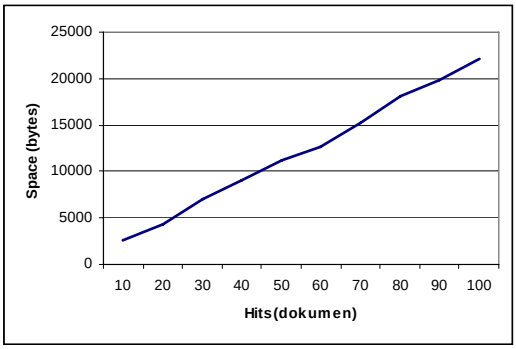


Gambar 4. Hasil pencarian tercluster dengan query 'asosia'

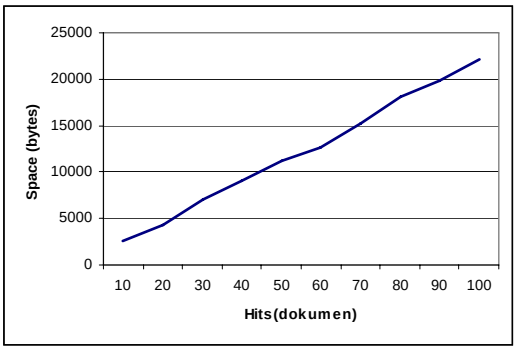
2. ‘P3L data mining utk mencari aturan-aturan **asosiasi** sesuai dgn permintaan pengguna’.
3. ‘Perancangan dan pembuatan perangkat lunak yang mengikuti pola distribusi poisson untuk pencarian kaidah **asosiasi** dan pola sekuensial’.
4. ‘Perancangan dan pembuatan perangkat lunak data mining untuk pencarian **kaidah asosiasi** dengan metode Bottom-Up’.
5. ‘Perancangan dan pembuatan perangkat lunak data mining untuk penggalian **kaidah asosiasi** menggunakan metode Hybrid’.

Uji coba yang kedua dilakukan dengan tujuan untuk mengetahui hubungan antara hits (jumlah dokumen hasil pencarian) dengan waktu konstruksi Suffix Cactus dan hubungan antara hits dengan ukuran Suffix cactus yang terbentuk.

Jumlah dokumen hasil pencarian bertambah secara konstan dari 10, 20, 30 sampai 100 dokumen.



Gambar 5. Grafik hubungan antara Hits dan Ukuran Suffix Cactus (*space*).



Gambar 6. Grafik hubungan antara Hits dengan Waktu Eksekusi.

Dari gambar 5 bisa kita lihat kecenderungan kebutuhan ruang memory (*space*) untuk membentuk Suffix Cactus terhadap *hits*. Terdapat kecenderungan *space* terhadap jumlah *hits* bersifat linear. Dari grafik ini pula dapat terlihat bahwa proses eksekusi dokumen hasil pencarian menjadi struktur Suffix Cactus tidak menambah kebutuhan *space* secara signifikan.

Hubungan antara waktu eksekusi untuk mengkonstruksi Suffix Cactus ditunjukkan pada Gambar 6. Bentuk grafik menggambarkan bahwa waktu eksekusi yang dibutuhkan akan bertambah seiring dengan bertambahnya dokumen hasil pencarian. Bentuk grafik berupa grafik yang eksponensial. Artinya peningkatan penambahan waktu eksekusi yang dibutuhkan selalu terjadi saat penambahan hits.

6. KESIMPULAN

Metode Suffix Cactus mampu menghasilkan cluster-cluster yang relatif membantu pengguna untuk mendapatkan informasi serta memberikan kategori browsing yang dinamis tergantung pada keyword yang diinputkan pengguna.

Metode Suffix Cactus Clustering mampu manghasilkan cluster dengan tingkat relevansi yang baik, mampu memberikan deskripsi cluster yang ringkas dan relatif membantu pengguna.

DAFTAR PUSTAKA

[1]. Juha Karkkainen, “Suffix Cactus : A Cross Between Suffix Tree and Suffix Array”, Department of Computer Science, University of Helsinki,Finland.

[2]. Oren Ali Zamir, “Clustering Web Documents : A Phrase-Based Methode for Grouping Search Engine Result”, PhD Thesis, University of Washington, 1999.

[3]. F.X. Arunanto, Fitrio Pakana, dan Aris Tjahyanto, “Perancangan Aplikasi Web-based Untuk Pencarian Dokumen Dengan Menggunakan Metode Suffix Tree Clustering (STC) Pada Result Set”, *Prosiding Seminar Nasional “Intelligent Technology and Its Applications (SITIA2001)”*, AI 1.1 - AI 1.4, 1 Mei 2001.

[4]. Yiming Yang,Jaime G.Carbonell, Rulf D. Brown, Thomas Pierce, Brian T. Achibald, Xin Liu. *“Learning Approaches for Detecting and Tracking News Event”*. IEEE Intelligent Systems ,Language Techlonolies Institute, Carbegie Mellon University, 1999.

[5]. M. Charikar, C.Chekuri, T.Feder, and R.Motwani, “Incremental Clustering and Dynamic Information Retrieval”, *Proceedings of the Twenty-Ninth Symposium on Theory of Computing*, 1997.

[6]. E. M. McCreight. “A Space-Economical Suffix Tree Construction Algorithm”, J. ACM, 1976.

[7]. Ricardo Baeza-Yates, Berthier Ribeiro-Neto, “*Modern Information Retrieval”*ACM Press,

New York, Addison Wasley Longman Limited, 1999.

[8]. William B.Frakes, Richardo Baeza-Yates, " *Information Retrieval Data Structures And Algorithm"*, Prentice-Hall International Edition, 1992.

[9]. Oren Eli Zamir and Oren Etzioni, "Web Document Clustering: A Feasibility Demonstration", *Proceeding of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM SIGIR'98, 1998.

[10]. Cutting D.R., karger, D.R, Pedersen, J.O and Tuckey, J.W.,"Scatter/Gather: A Cluster-Based Approach to Browsing Large Document Collections", *Proceeding of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM SIGIR'92, halaman 318-329, 1992.

[11]. Setiono, Ari Novan "*Implementasi Aplikasi Information Retrieval* untuk Pendeteksian dan Klasifikasi Berita Kejadian Berbahasa Indonesia Berbasis Web", Tugas Akhir, Teknik Informatika, Institut Teknologi Sepuluh Nopember Surabaya, 2002.