Pengambilan Inf (2006) 9:499–
517 DOI 10.1007/s10791-006-6614-y

Solusi berbasis TSP dan cluster untuk penugasan ulang pengidentifikasi dokumen

Roi Blanco · **Alvaro Barreiro**

Diterima: 1 Juli 2005 / Direvisi: 9 Desember 2005 / Diterima: 9 Desember 2005

Abstrak Studi terbaru menunjukkan bahwa ukuran *File Terbalik* (IF) dapat dikurangi dengan menetapkan ulang pengidentifikasi dokumen dari koleksi asli, karena hal ini akan menurunkan jarak antara posisi dokumen yang terkait dengan satu istilah. Skema pengkodean bit variabel dapat memanfaatkan pengurangan kesenjangan rata-rata dan mengurangi jumlah total bit per penunjuk dokumen. Makalah ini menyajikan solusi efisien untuk masalah penugasan ulang, yang terdiri dari pengurangan dimensi data input menggunakan transformasi SVD, serta mempertimbangkannya sebagai *Traveling Salesman Problem* (TSP). Kami juga menyajikan beberapa solusi efisien berdasarkan clustering. Terakhir, kami menggabungkan TSP dan strategi pengelompokan untuk menyusun ulang pengidentifikasi dokumen. Kami menyajikan pengujian eksperimental dan hasil kinerja dalam dua koleksi teks TREC, memperoleh rasio kompresi yang baik dengan waktu berjalan yang rendah, dan memajukan kemungkinan memperoleh solusi terukur untuk koleksi web berdasarkan teknik yang disajikan di sini.

Kata Kunci Penugasan kembali pengidentifikasi dokumen. SVD. Pengindeksan. Kompresi . TSP. Kekelompokan

1. Perkenalan

Sistem Pengambilan Informasi (IR) skala besar memerlukan mekanisme pengindeksan untuk pengambilan yang efisien. Struktur data yang paling luas digunakan adalah *file terbalik* (IF). IF ini merupakan representasi penelusuran dari kumpulan dokumen asli, yang disusun dalam *daftar posting.* Setiap entri dalam file terbalik berisi informasi tentang satu istilah dalam kumpulan dokumen. Format daftar postingan mencerminkan rincian file yang dibalik, menangani dokumen dan posisi mana istilah tersebut muncul. Dalam pekerjaan ini kami mengasumsikan tingkat dokumen

R.Blanco () . A.Barreiro IRLab.
Departemen Ilmu Komputer, Universitas Corunna, Spanyol email: rblanco@udc.es

A. Email
Barreiro: barreiro@udc.es

granularitas, oleh karena itu daftar posting untuk istilah *ti* adalah:

$$< ti ; baik ; \text{d1, d2,...,} d f ti >, di < \text{djÿi} < j \quad (1)$$

dimana *fti* adalah singkatan dari frekuensi istilah *ti* (jumlah dokumen yang memuat *ti),* dan *di* adalah pengidentifikasi dokumen. Sesuai dengan notasinya, pengidentifikasi dokumen diurutkan.

Karena struktur ini memerlukan ruang penyimpanan yang besar, daftar posting biasanya dikompresi. Beberapa penelitian telah membahas pengkodean pengidentifikasi dokumen yang efisien yang terkandung dalam setiap entri (Witten et al., 1999). Daftar posting disimpan sebagai urutan perbedaan antara pengidentifikasi dokumen yang berurutan *(d-gap).* Metode ini meningkatkan kompresi, karena skema pengkodean dengan panjang variabel merepresentasikan bilangan bulat kecil dengan bit lebih sedikit dibandingkan bilangan bulat besar. D-gap yang kecil lebih sering terjadi daripada yang besar, sehingga file yang diinversi dapat dikompresi secara efisien. Penelitian terbaru telah mencoba untuk meningkatkan jumlah d-gap kecil dengan menyusun ulang pengidentifikasi dokumen, dengan harapan bahwa panjang rata-rata dari kode d-gap akan diturunkan.

Penugasan ulang pengidentifikasi dokumen telah diatasi dengan dua strategi berbeda: memisahkan koleksi dengan algoritma partisi grafik atau pengelompokan dan memperoleh tatanan baru yang mengeksploitasi lokalitas (Blandford dan Blelloch, 2002; Silvestri et al., 2004), dan mempertimbangkan tugas sebagai masalah tata letak grafik (Shieh et al., 2003). Dalam karya yang disajikan dalam Blanco dan Barreiro (2005), heuristik yang menganggap masalah sebagai *Traveling Salesman Prob-lem* (TSP) diperkenalkan dalam Shieh et al. (2003) mengizinkan solusi penugasan ulang yang efisien setelah mengurangi dimensi data masukan menggunakan transformasi *Singular Value Decomposition* (SVD). Strategi berbasis TSP menganggap penugasan ulang pengidentifikasi dokumen sebagai masalah traversal grafik dan menemukan jalur sesuai dengan minimum global. Meskipun strategi ini tidak secara eksplisit didasarkan pada lokalitas, namun dalam praktiknya telah memperoleh hasil yang sangat menarik.

Seperti yang ditunjukkan nanti di Bagian 2, jalur ini adalah jalur yang memaksimalkan perpotongan bitwise dari vektor dokumen. Pada Bagian 6 kami memberikan penjelasan tentang perilaku heuristik TSP yang baik untuk masalah ini.

Alasan utama mengapa reduksi SVD berguna dalam konteks ini adalah karena teknik TSP perlu menghitung nilai kesamaan antar dokumen untuk memilih tepi yang akan menghubungkan node di jalur akhir. Dengan demikian, komputasi SVD menangani masalah efisiensi yang diperlukan mengingat dimensi ruang istilah berdasarkan dokumen, sekaligus menjadi pilihan terbaik untuk transformasi matriks, karena memperoleh ukuran kesamaan terbaik dalam ruang berdimensi tereduksi.

Makalah ini memperluas pekerjaan yang disajikan dalam Blanco dan Barreiro (2005) dengan meninjau dan menerapkan kembali dua algoritma pengelompokan yang sebelumnya disesuaikan dengan masalah penugasan kembali (Silvestri et al., 2004), untuk mengevaluasi dua pendekatan berbeda dalam kondisi yang sama. Dalam proposal yang disampaikan dalam Silvestri et al. (2004) penugasan pengidentifikasi dokumen dilakukan *dengan cepat,* yaitu tanpa adanya file terbalik yang dibuat sebelumnya.
Faktanya, inilah alasan utama mengapa dasar evaluasi pendekatan ini adalah urutan acak dan bukan urutan koleksi asli. Dalam pekerjaan kami, kami mengukur perbaikan setelah penugasan ulang pengidentifikasi dokumen sehubungan dengan urutan acak dan koleksi identitas (asli). Kemudian, kami menyajikan teknik baru yang menggabungkan kedua strategi tersebut sedemikian rupa sehingga tradeoff terbaik antara kompresi dan efisiensi dapat dicapai.

Sisa makalah ini disusun sebagai berikut. Bagian 2 menjelaskan solusi berbasis TSP secara mendalam. Bagian 3 menunjukkan cara mengurangi dimensi matriks kesamaan dokumen dengan menghitung Dekomposisi Nilai Singularnya dan bagaimana kami menerapkan hasil ini pada masalah penugasan ulang. Bagian 4 menerapkan teknik SVD lagi, tetapi setelah dilakukan pembagian

masalah awal menjadi submasalah. Kami menunda pengenalan solusi pengelompokan dan teknik yang menggabungkan TSP dan pengelompokan hingga Bagian 5, karena penerapan, eksperimen, dan hasil dari masing-masing teknik disertakan dalam bagian terkait. Makalah ini dilanjutkan dengan diskusi dan pekerjaan masa depan di Bagian 6, dan diakhiri dengan bagian kesimpulan.

2. Pendekatan TSP terhadap masalah penugasan kembali pengidentifikasi dokumen

Mengingat bahwa kami akan mengilustrasikan penggunaan teknik reduksi dimensi untuk penugasan ulang dokumen dengan solusi berbasis TSP, kami meninjau secara singkat pekerjaan di Shieh dkk. (2003).

2.1. Dokumen mengidentifikasi masalah penugasan kembali sebagai TSP

File terbalik dapat dilihat sebagai kumpulan daftar posting. Setiap daftar berisi informasi untuk satu istilah yang muncul dalam kumpulan dokumen, dinyatakan sebagai urutan d-gaps *Gt* = {g1,..., *g* ft} yang dikodekan. Masalah penugasan kembali dokumen mencoba mencari fungsi bijektif *f* itu

– memetakan setiap pengidentifikasi dokumen ke dalam pengidentifikasi baru dalam kisaran
[1 ... d] – meminimalkan biaya dalam bit pengkodean daftar posting

Kesamaan antar dokumen didefinisikan sebagai jumlah suku yang sama, dan dipelihara dalam matriks kesamaan *Sim,* dimana *Simi j* mewakili kesamaan antara dokumen *i* dan dokumen j.

Shieh dkk. (2003) mengusulkan strategi pengurangan kesenjangan yang didasarkan pada transformasi masalah menjadi *Travelling Salesman Problem* (TSP). TSP dinyatakan sebagai berikut: diberikan graf berbobot *G* = *(V,* E) dimana e(vi, *vj)* adalah bobot sisi dari *vi* untuk mencari lintasan minimal *P* = {v1, v2,...,vn } berisi semua titik di V, seperti if *vj , P* = {v e(v 1, v 2,...,v n} adalah jalur lain di G, $\sum_{saya=2}^{N}$ e(vi, viÿ1) ÿ $\sum_{saya=2}^{N}$ v sayaÿ1).

Mengingat *Sim* sebagai matriks ketetanggaan berbobot, dimungkinkan untuk membuat Grafik Kemiripan Dokumen (DSG) yang mengungkapkan kesamaan antar dokumen. Grafik ini dapat dilintasi dengan strategi pengurangan kesenjangan berdasarkan faktor kesamaan antar dokumen. Idenya adalah untuk menetapkan pengidentifikasi dokumen yang mirip dengan dokumen serupa, karena hal ini kemungkinan akan mengurangi kesenjangan-d dalam postingan istilah umum. Masalah traversing ini dapat diubah menjadi TSP hanya dengan mempertimbangkan komplemen kemiripan dengan bobot pada TSP.
Solusi dari TSP adalah jalur yang meminimalkan jumlah jarak antar dokumen, oleh karena itu algoritma ini merupakan strategi yang tepat untuk masalah penugasan ulang dokumen.

2.2. Perkiraan heuristik

TSP adalah masalah NP-lengkap, sehingga beberapa perkiraan heuristik waktu polinomial dimodifikasi untuk masalah penugasan ulang. Algoritma ini diklasifikasikan menjadi algoritma serakah dan algoritma spanning tree. Kami menguji perkiraan dimensi rendah kami dengan algoritma Greedy-NN.

Saat mendeskripsikan pendekatan heuristik, kami menganggap TSP sebagai jalur yang memaksimalkan jumlah kesamaan antara dokumen yang berurutan. Greedy-NN (Nearest Neighbor) memperluas jalur dengan menambahkan simpul terdekat ke ekor jalur saat ini. Pada setiap iterasi algoritma menambahkan simpul (dokumen) baru, dengan memilih simpul yang paling mirip dengannya

Tabel 1 Statistik murni Pendekatan TSP pada dua koleksi TREC yang dilaporkan oleh Shieh dkk. (2003)

Koleksi	FBI	LATimes
Ukuran koleksi	470MB	475MB
Jumlah istilah yang berbeda	209.782	167.805
Jumlah dokumen berbeda	130.471	131.896
Biaya sementara	19 jam 37	23 jam 16
Biaya ruang	2,10GB	2,17GB

yang terakhir di jalan. Perkiraan ini memakan banyak waktu. Setiap titik disisipkan hanya sekali di jalur *P* dan pada iterasi *i* algoritma melakukan perbandingan *d* ÿ *i*

Algoritma Greedy-NN

1: **Masukan:**
Grafik *G*
Himpunan Puncak *V*
Tepi tertimbang mengatur *E*
2: **Keluaran:**
Jalur global *P* yang memaksimalkan kesamaan antar simpul
3: Pilih sisi e(vi, *vj)* ÿ *E* dengan bobot terbesar;
4: Tambahkan *vi* dan *vj* ke P;
5: *terakhir* ÿ *vj ;*
6: **sementara** *(*|*P*| = |V|) **lakukan**
7: Pilih *vk* ÿ *V* dan *vk* ÿ/ *P* sehingga *e(vlast, vk)* maksimal;
8: Tambahkan *vk* ke P;
9: *vlast* ÿ *vk ;*
10: **berakhir sementara**
11: **kembalikan P;**

(dokumen yang tersisa) melibatkan term size *t* dari kedua dokumen. Oleh karena itu secara keseluruhan kompleksitasnya adalah O(d2t).

2.3. Pertimbangan implementasi

Perkiraan TSP untuk masalah penugasan kembali pengidentifikasi dievaluasi di Shieh dkk. (2003). Solusi ini menunjukkan peningkatan yang signifikan dalam rasio kompresi, meskipun hal ini menghadirkan beberapa tantangan desain dan hasil kinerja ruang dan waktu yang buruk.

Pertama, pendekatan ini memerlukan ruang yang besar. Matriks kemiripannya simetris *(Simi j = Simji)* dan elemen-elemen pada diagonalnya tidak relevan, sehingga mudah untuk membuktikannya kita perlu menyimpan $2\frac{d(d\ddot{y}1)}{\text{jumlah yang}}$ *penunjuk kesamaan* (O(d2)). Bahkan dengan skema pengkodean yang sesuai ini tidak dapat dikelola, sehingga teknik partisi matriks harus dikembangkan.

Kedua, membangun matriks ini bisa sangat mahal jika tidak sesuai dengan memori masing-masing pembaruan harus mengakses disk dua kali, menyebabkan penundaan yang besar.

Hasil eksperimen disajikan untuk dua koleksi berukuran sedang (FBIS dan LATimes di TREC disk 5), untuk membuktikan efektivitas mekanisme ini. Tes-tes ini dirangkum pada Tabel 1.

Penting untuk dicatat bahwa penelitian di Shieh dkk. (2003) memberikan grafik batang yang menunjukkan perkiraan perolehan satu bit per celah saat menugaskan ulang dengan Greedy-NN untuk delta dan pengkodean gamma. Biaya temporal mencakup proses membangun matriks kesamaan,

menjalankan algoritma serakah dan mengompresi ulang file yang dibalik. Namun, hasilnya terlihat bahwa pendekatan TSP penuh ini mungkin tidak dapat diterima untuk koleksi yang sangat besar, karena memerlukan waktu 23 jam dan 2,17 GB untuk memproses koleksi 475 MB.

3. Penugasan ulang pengidentifikasi dokumen dengan pengurangan dimensi dan TSP strategi

Strategi TSP mencapai rasio kompresi yang sangat baik. Untuk alasan ini, kami mengusulkan yang baru pendekatan berdasarkan reduksi dimensi di mana algoritma TSP dapat beroperasi secara efisien. Teknik ini didasarkan pada penerapan transformasi SVD pada data masukan, untuk mendapatkan representasi baru dari struktur data yang memungkinkan penataan ulang pengidentifikasi dokumen dalam ruang yang dikurangi dimensinya. Setelah pesanan baru diperoleh, daftar posting dalam file terbalik dikompresi ulang, menggunakan pengidentifikasi dokumen yang diberikan oleh fungsi penugasan. Perlu diperhatikan sifat berbeda dari aplikasi SVD ini dalam Pengambilan Informasi sehubungan dengan penggunaan IR SVD lainnya, seperti pengambilan LSI model.

3.1. Dekomposisi Nilai Tunggal

Dekomposisi Nilai Singular (SVD) adalah teknik matematika terkenal yang digunakan secara luas berbagai bidang. Ini digunakan untuk menguraikan matriks persegi panjang sembarang menjadi tiga matriks mengandung vektor tunggal dan nilai tunggal. Matriks ini menunjukkan perincian dari hubungan asli menjadi faktor-faktor yang bebas linier. Teknik SVD digunakan sebagai dasar matematika model IR Latent Semantic Indexing (LSI) (Deerwester et al., 1990).

Secara analitis, kita mulai dengan X, matriks syarat dan dokumen berukuran $t \times d$. Kemudian, menerapkan SVD X didekomposisi menjadi tiga matriks:

$$X = \text{T0 S0D}_0 \quad (2)$$

T0 dan D0 memiliki kolom ortonormal, dan S0 diagonal dan, berdasarkan konvensi, $s_{ii} \ddot{y} 0$ dan $s_{ii} \ddot{y} s_j$ jÿi ÿ j. T0 adalah matriks $t \times m$, S0 adalah $m \times m$ dan D_0 adalah $m \times d$ di mana m adalah pangkatnya dari X. Namun dimungkinkan untuk memperoleh perkiraan peringkat k dari matriks asli X dengan menjaga k nilai terbesar di S0 dan menyetel sisanya ke nol, sehingga diperoleh matriks S dengan dimensi $k \times k$. Karena S adalah matriks diagonal dengan k nilai bukan nol, maka kolom yang sesuai dari T0 dan D_0 dapat dihapus untuk mendapatkan T, berukuran $t \times$ k, dan D, berukuran $k \times$ d, masing-masing.

Dengan cara ini kita dapat memperoleh $\hat{X}$, yang merupakan perkiraan peringkat k tereduksi dari X:

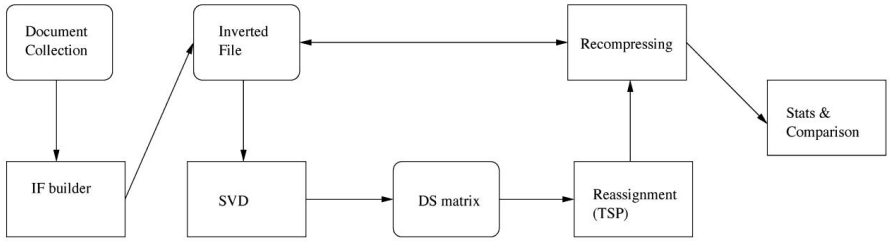
$$X \ddot{y} \hat{X} = TSD \quad (3)$$

$\hat{X}$ adalah perkiraan peringkat k terdekat dari X dalam kaitannya dengan norma Euclidean atau Frobenious, yaitu matriks yang meminimalkan $||X \ddot{y} \hat{X}||_2$ dimana $|| \cdot ||_2$ adalah norma yang terlibat.

Baris ke-i dari DS memberikan representasi dokumen i dalam ruang-k tereduksi dan matriks kemiripan (X) didekati k dengan $(\hat{X})$:

$$(X) \ddot{y} (\hat{X}) = \hat{X}\hat{X} = \text{DS2D}, \quad (4)$$

dimana $\hat{X}$ adalah matriks yang ditransposisikan dari $\hat{X}$ dan D adalah transposisi dari D.



Gambar 1 Diagram blok untuk sistem pengindeksan dan penugasan ulang

Jika Dd×k = *{zi* j} dan *{si}* adalah himpunan elemen diagonal S, mudah untuk membuktikannya

$$(\hat{X})_{ij} = \sum_{\ddot{y}=0}^{k\ddot{y}1} z_{i\ddot{y}} z_{j\ddot{y}} \ddot{y} \, s_{\ddot{y}}^2 \qquad (5)$$

Oleh karena itu, *(Xˆ)ij* dapat dihitung hanya dengan menyimpan *k* elemen dari *{si}* himpunan dan matriks *d* × *k* D, alih-alih menghitung dan menulis matriks pangkat penuh *(X)d×d* .

Output dari SVD dari X, Xˆ telah digunakan dalam perhitungan *(Xˆ)* = Xˆ · Xˆ Hasil yang sama dapat diperoleh dengan menghitung SVD dari *(X)* = *X* · *X* karena properti keunikan SVD (Bartell et al., 1992). Karena SVD menghitung peringkat *k* terbaik aproksimasi, terbukti bahwa aproksimasi rank *k* terbaik dari *(X)* diperoleh mulai dari *X* dan tanpa perlu komputasi (X).

3.2. SVD dalam masalah penugasan kembali dokumen

Gambar 1 menjelaskan sistem yang dibangun untuk menguji pendekatan ini. Mekanisme pembuat file terbalik mengeluarkan matriks data *X* ke modul SVD. Modul ini menghasilkan matriks Dd×k dan Sk×k yang memungkinkan penghitungan *(Xˆ),* sehingga tidak perlu lagi menyimpan matriks kesamaan *(X)d×d* . Modul penugasan ulang menggunakan matriks keluaran SVD untuk menghitung pendekatan TSP yang dijelaskan dalam Bagian 2.2. Karena *k* adalah faktor konstan, kita dapat menyimpulkan bahwa penggunaan ruang dari algoritme sekarang adalah *O(d)* , yaitu linier dalam ukuran koleksi dan tidak bergantung pada ukuran dokumen. Output dari modul penugasan ulang TSP digunakan oleh file terbalik program pengodean ulang yang memanfaatkan lokalitas baru dokumen untuk meningkatkan kesenjangan-d kompresi. Akhirnya, beberapa informasi statis diperhitungkan untuk membuat perbandingan yang sesuai antara rasio kompresi yang dicapai oleh pengkodean asli dan yang diperoleh.
setelah penugasan kembali.

Langkah-langkah yang dirangkum dirinci berikutnya, di mana SVDGreedyNN adalah versi modifikasi dari algoritma Greedy-NN yang menggunakan matriks *(Xˆ)* untuk menghitung ukuran kesamaan antara simpul *vi* dan *vj* .

Penugasan ulang pengidentifikasi dokumen berbasis SVD

1: **Masukan:**
Pengumpulan *DokumenC*
Parameter *k* SVD
2: **Keluaran:**
File Terbalik yang disusun ulang
3: *Indeks asli* ÿ **createIndex(C);** 4: *matriks tereduksi* ÿ *SVD (Indeks asli,* **k);** 5: *documentOrder* ÿ *SVDGreedyNN(reduksiMatrix);* 6: *indeks baru* ÿ **kompres ulang (C,** *documentOrder);* 7: *kembali (Indeks baru);*

Perbedaan utama dalam model ini adalah menghitung kesamaan antara dua dokumen *di* dan *dj* memerlukan *k* operasi (kÿ1 =*0(DS)iÿ* (DS)jÿ) dan menyimpan*k* pointer nyata per dokumen, sehingga totalnya adalah *k* × *d* untuk matriks penuh. Representasi ini dapat masuk dengan lancar ke dalam memori dengan menyesuaikan parameter *k* dan menggunakan ruang yang jauh lebih sedikit dibandingkan matriks *d* × *d* asli . Selain itu, penggunaan ruang dapat dihitung terlebih dahulu sehingga algoritma terukur yang sesuai dapat dikembangkan dengan mudah. Mengingat 32 bit per float (bilangan real), implementasi kami menggunakan memori utama 4 × *k* × *d* byte.

Salah satu hal yang perlu dipertimbangkan adalah heuristik untuk memilih node awal pada algoritma Greedy-NN yang juga digunakan untuk menyelesaikan TSP. Algoritma (Bagian 2.2) pertama-tama memilih sisi *(vi, vj)* yang mempunyai nilai maksimum. Hal ini melibatkan penghitungan kesamaan untuk setiap pasangan dokumen *(di,* dj). Dalam pendekatan kami, memilih node pertama dengan cara ini melibatkan lebih banyak operasi daripada algoritma itu sendiri. Oleh karena itu, kami mengusulkan heuristik yang lebih hemat waktu, yang terdiri dari perhitungan, setelah reduksi dimensi, masing-masing *(di, di)* kesamaan diri dan memilih dokumen (node) dengan nilai terbesar.

3.3. Eksperimen dan hasil

Kami melakukan beberapa percobaan untuk menguji pendekatan dimensi rendah pada dua koleksi dokumen TREC yang dijelaskan pada Tabel 1. Koleksi ini tidak diproses sebelumnya, sehingga pengindeksan dan penyusunan ulang menyertakan kata-kata berhenti dan istilah-istilah tidak berasal. Mesin yang digunakan adalah AMD 2,5 GHz dengan disk ATA 40 GB dan memori utama 1 GB, menjalankan OS Linux. File indeks asli dibuat dengan MG4J dari Universitas Milan, implementasi teknik pengindeksan dan kompresi Java gratis yang dijelaskan dalam Witten et al. (1999) dan awalnya diimplementasikan dalam perangkat lunak MG (http://www.cs.mu.oz.au/mg/). Untuk modul SVD kami menggunakan SVDLIBC (http://tedlab.mit.edu/ÿ dr/SVDLIBC/), perpustakaan C berdasarkan perpustakaan SVDPACKC. Kami menulis perangkat lunak penugasan ulang, pengodean ulang, dan statistik di Java. Perlu diperhatikan bahwa kami perlu memodifikasi perangkat lunak MG4J untuk mengeluarkan data langsung ke modul SVDLIBC. Beberapa modifikasi juga dilakukan yang memungkinkan kami melakukan pengkodean
penunjuk dokumen dengan pengkodean interpolatif.

Eksperimen pertama menilai kinerja sistem dengan algoritma Greedy-NN, dalam hal rata-rata bit per penunjuk dokumen terkompresi (d-gap). Koleksi dokumen dibalik, IF dimasukkan ke modul SVD dan program menghitung Greedy-NN dalam dimensi tereduksi untuk tugas penugasan ulang. Setelah menyusun ulang koleksi, file yang dibalik dikompresi ulang. Perangkat lunak ini mengukur rata-rata bit per celah dalam file yang dibalik, sebelum dan sesudah menyusun ulang dan mengompresi ulang, yang mencerminkan

jumlah kompresi yang diperoleh dengan menyusun ulang koleksi dokumen. Kami menjalankan beberapa pengujian dengan memvariasikan parameter berikut:

– parameter *k* yang mencerminkan reduksi dimensi yang diinginkan – skema pengkodean untuk penunjuk dokumen: pengkodean delta, pengkodean gamma, pengkodean golomb atau pengkodean interpolatif (Moffat dan Turpin, 2002).

Hasil terbaik diperoleh dengan mempertimbangkan *X* sebagai matriks biner dalam proses penugasan kembali. Unsur *X* mewakili ada atau tidaknya suatu istilah dalam suatu dokumen tertentu. Modul kompresi ulang bertindak atas file indeks asli yang berisi frekuensi istilah dalam dokumen dan frekuensi istilah dalam koleksi. Hasil diberikan dalam bit per celah dokumen karena ini merupakan ukuran yang independen terhadap opsi pengindeksan ini. Sebagaimana dinyatakan dalam Bagian 3, penggunaan memori menghasilkan 4 × *d* × *k* byte, secara konkret 0,497707 × *k* MB untuk FBIS dan 0,503143 × *k* MB untuk LATimes (untuk *k* = 200 kurang dari 101 MB di kedua koleksi).

Kami telah bereksperimen dengan berbagai algoritma pengkodean yang telah berhasil digunakan untuk kompresi file terbalik. Algoritme ini mencakup beberapa kategori berbeda untuk metode pengkodean statis (Moffat dan Turpin, 2002): dua metode global yang tidak diparameterisasi (gamma ÿ dan delta ÿ (Elias, 1975)), satu metode berparameter global (kode Golomb (Golomb, 1966)), dan satu peka terhadap konteks lokal (pengkodean interpolatif (Moffat dan Stuiver, 2000)).

Kode Elias delta dan gamma (Elias, 1975) tidak memperhitungkan data apa pun dari kumpulan dokumen, karena keduanya menghasilkan representasi yang sama untuk bilangan bulat *x* berapa pun frekuensinya. Untuk kode gamma, probabilitas tersirat untuk celah dengan panjang *x* adalah

$$\Pr(x) = ; 2x\ 2^{\frac{1}{}} \qquad (6)$$

untuk kasus kode delta kemungkinannya adalah

$$\Pr(x) = . \ (7)^{\frac{1}{}} \text{ 2x(catatan x)2}$$

Panjang kode terkait untuk *x* masing-masing adalah 1 + 2log *x* dan 1 + 2log log x+log *x* . Di sisi lain, pengkodean Golomb global (Golomb, 1966) memanfaatkan kepadatan pointer dalam file terbalik dan bergantung pada parameter b. Pengkodean Golomb optimal jika probabilitas tersirat mengikuti distribusi geometri:

$$\Pr(x) = (1 \text{ ÿ } p)^{(x\text{ÿ}1)} \ P. \ (8)$$

Panjang kode untuk bilangan bulat *x* adalah *(x* ÿ 1)/b+logb bit. Parameter *b* dapat dipilih sehingga jumlah rata-rata bit untuk pengkodean file yang dibalik diminimalkan (Gallager dan van Voorhis 1975) dan bergantung pada distribusi geometri. Terakhir, pengkodean interpolatif (Moffat dan Stuiver, 2000) adalah model peka konteks yang memperhitungkan properti pengelompokan koleksi. Perilaku kode interpolatif hampir sama efisiennya dengan kode Golomb dalam kasus terburuk, meskipun hal ini dapat menghasilkan keuntungan kompresi yang lebih besar ketika menangani kumpulan bilangan bulat yang berdekatan.

Tabel 2 dan 3 menunjukkan hasil berbagai skema pengkodean. Pada baris yang diberi label dengan masing-masing metode pengkodean, kolom mengacu pada hasil bit per celah dokumen untuk: penugasan ulang secara acak, pengidentifikasi dokumen asli, dan penugasan ulang setelah mengurangi dimensi dengan nilai *k* yang berbeda . Sebenarnya, titik awal default untuk penataan ulang apa pun adalah tatanan yang sudah ada sebelumnya, yang sudah menyertakan elemen pengelompokan karena

Tabel 2 Hasil bit LATimes per gap dan waktu eksekusi

			k						
	Acak Asli 200			100	50	20	10	5	1
Gamma	8.15	7.77	6.71	6.75	6.79	6.89	6.99	7.13	7.44
Delta 7.65 Interpolatif		7.25	6.29	6.36	6.39	6.48	6.57	6.73	7.02
6.08		5.88	5.25	5.26	5.28	5.29	5.33	5.44	5.57
Golomb	5.82	5.81	5.79	5.79	5.79	5.79	5.79	5.79	5.79
SVD	–	–	42 31	20 37	11 25 4 05 2 33 2h15 58 18			1 55	1 09
Susun ulang	–	–	8 jam 33	4 jam 24	32 05			18 31 7 30	

Tabel 3 Hasil bit FBIS per celah dan waktu eksekusi

			k						
	Acak Asli 200			100	50	20	10	5	1
Gamma	7.84	6.74	6.20	6.24	6.31	6.46	6.63	6.81	7.07
Delta	7.35	6.35	5.80	5.86	5.92	6.07	6.23	6.42	6.69
Interpolatif 5.83 Golomb		5.25	4,98	4,99	5.01	5.06	5.17	5.21	5.33
5.61		5.60	5,59	5,59	5.59	5.59	5.59	5.59	5.59
SVD	–	–	34 58	15 01 5 42 3 18			2 09	1 33	58
Susun ulang	–	–	08.30	4 jam 20	1 jam 48 58 20		31 13	17 55	7 05

kronologis dokumen. Namun pada penelitian sebelumnya (Blandford dan Blelloch, 2002; Silvestri dkk., 2004), penulis menggunakan penugasan ulang secara acak sebagai dasar, dan untuk masalah perbandingan, kolom ini juga disertakan di sini. Dengan menetapkan nilai ke *k* yang mirip dengan yang digunakan dalam pengambilan (Dumais, 1994), algoritma dimensi rendah beroperasi dengan keuntungan yang menghasilkan perbaikan dalam bit per gap. Seperti yang diharapkan, metode ini berperilaku lebih baik nilai *k* meningkat. Selain itu, tabel tersebut tampaknya menunjukkan perilaku asimtotik. Dengan *k* = 200, untuk koleksi LATimes (koleksi FBIS) kami mencapai peningkatan rasio kompresi sebesar 13,65% (8,02%) sehubungan dengan urutan pengidentifikasi dokumen asli dengan pengkodean gamma, 13,24% (8,66%) untuk pengkodean delta, dan 11,32% (5,15%) untuk pengkodean interpolatif. Nilai tersebut masing-masing adalah 17,67% (21,92%), 17,80% (21,10%) dan 13,66% (14,58%). untuk koleksi dan tiga skema pengkodean, sehubungan dengan penugasan ulang secara acak. Menghitung TSP Greedy-NN dengan perkiraan ruang tereduksi *(X^)* menghasilkan rasio kompresi yang layak dalam setiap kasus. Keuntungan dalam pengumpulan FBIS lebih buruk daripada yang ada di LATimes, walaupun dimulai dari urutan acak hasilnya terbalik. Ini adalah perilaku yang diharapkan jika koleksi FBIS memperlihatkan dokumen asli yang lebih baik memesan. Satu hal yang perlu diperhatikan adalah bahwa bahkan dalam kasus pengkodean interpolatif, di mana titik awalnya jauh lebih baik, metode ini mampu menghasilkan keuntungan dalam bit per celah dokumen. Mengenai rasio kompresi dengan pengkodean Golomb, ini memberikan kinerja terbaik untuk penugasan acak, dan jelas bahwa ini tidak memanfaatkan lokalitas dokumen. Alasan perilaku ini adalah karena kode Golomb tidak terpengaruh oleh kemiringan distribusi dokumen, karena menjadikan distribusi geometris (Moffat dan Turpin, 2002). Namun, pengkodean interpolatif dengan penyusunan ulang adalah opsi yang menghasilkan kompresi terbaik hasil.

Pengujian kami tidak mencakup penghitungan solusi dimensi penuh seperti yang disajikan dalam Shieh dkk. (2003), karena memerlukan pengembangan teknik partisi matriks

dan membaca/menulis sebagian, yang merupakan tugas yang ingin kita hindari. Shieh dkk. (2003) memberikan hasil grafik batang untuk pengkodean gamma dan delta dalam koleksi LATimes dan FBIS. Namun, nilai kompresi yang tepat bergantung pada perangkat lunak pengindeksan dan opsi pengindeksan tertentu. Informasi ini tidak diberikan secara eksplisit, sehingga tidak mungkin dilakukan membuat perbandingan yang tepat antara hasil dimensi penuh yang dipublikasikan dan dimensi k solusi.

Dua baris terakhir Tabel 2 dan 3 dikhususkan untuk waktu berjalan. Pengukuran waktu dibagi menjadi tiga bagian: konstruksi file terbalik, waktu berjalan SVD dan penataan ulang dan waktu kompresi ulang. Waktu yang diberikan hanya untuk pengkodean delta, sebagaimana tabel menggambarkan waktu variasi dalam parameter *k* . Varians waktu akibat k, hanya bergantung pada SVD dan waktu penyusunan ulang, jadi waktu dengan pengkodean delta saja yang cocok untuk tujuan ini (waktu kompresi ulang sekitar 16 detik untuk kedua koleksi). Saat sistem dibangun modul yang berbeda, bagian perangkat lunak yang berbeda menggunakan banyak waktu transfer I/O sementara, yang juga diukur, sehingga hasilnya diberikan dalam *waktu yang telah berlalu.* Inversi membutuhkan waktu 5 20 untuk koleksi FBIS dan 6 03 untuk koleksi LATimes dan tidak ditampilkan di tabel berkah. Meskipun perangkat lunak SVD berkinerja baik untuk koleksi dan nilai *k* yang digunakan, namun Waktu berjalan algoritma serakah TSP masih meningkat ke nilai yang tinggi. Oleh karena itu, kami menyimpulkan bahwa memang demikian mungkin untuk mencapai rasio kompresi yang baik dengan kinerja waktu yang wajar dengan kami teknik. Di sisa makalah ini, kami akan menunjukkan hasil untuk satu metode kompresi, karena eksperimen ditujukan untuk mengklarifikasi apakah varian penataan ulang yang berbeda memberikan hasil yang lebih baik nilai bit per gap, beserta biaya temporalnya masing-masing, tanpa berfokus pada pengkodean metode.

4. Algoritma blok-c

Untuk lebih memanfaatkan keuntungan dari strategi TSP dan pengurangan dimensi, kami mengembangkan algoritma baru yang sederhana berdasarkan pembagian masalah asli di *c* submasalah, selanjutnya c-GreedyNN. Nanti kita bisa membandingkan kinerjanya antara algoritma ini dan algoritma lainnya berdasarkan partisi koleksi.

Algoritma c-GreedyNN beroperasi sebagai berikut: pertama, ia membagi matriks *DS* (yang mewakili kesamaan dokumen di *k* spasi) di *c* blok masing-masing dokumen [d/c] . Kemudian, setiap blok disusun ulang dengan menjalankan algoritma serakah. Akhirnya, ada perintah blok diputuskan dengan menjalankan serakah lain dengan dokumen *c ,* masing-masing dipilih dari yang berbeda blok. Untuk penjelasan yang lebih sederhana, kami menganggap *d* merupakan kelipatan eksak dari c. Secara analitis, itu Greedy-NN setelah reduksi dimensi melakukan *d* perbandingan untuk memilih dokumen pertama, dan $d\frac{(d-1)}{2}$ untuk menyusun ulang, menghasilkan $\frac{D}{2}$ perbandingan *(d* + 1) yang masing-masing melibatkan *k* perkalian. Pendekatan baru memilih perwakilan blok *c* dan kemudian melakukan proses serakah *c* dengan d/c dokumen, menghasilkan *d* + $c\left(\frac{\frac{D}{c}\left(\frac{D}{c}-1\right)}{2}\right) = \frac{D}{2}\left(\frac{D}{C}+1\right)$ perbandingan, jadi jumlah keseluruhan operasi dikurangi dengan faktor 1/c. Hasil percobaan dengan nilai bilangan yang berbeda blok *c* disajikan pada Tabel 4 dan 5.

Hasil disediakan untuk koleksi LATimes dan FBIS, *k* = 200 dan pengkodean delta. Tabel 4 dan 5 juga menunjukkan bahwa faktor kompresi meningkat seiring dengan jumlah blok menurun, dengan nilai sasaran 6,29 (5,80) untuk koleksi LATimes (FBIS), yaitu nilai mempertimbangkan matriks sebagai satu blok tunggal.

Algoritme ini efisien dalam waktu berjalan, seperti yang diharapkan dari bentuk analisisnya memberikan nilai kompresi yang dapat diterima. Kami menjalankan semua percobaan dengan *k* = 200, dan terbukti untuk mendorong hasil performa terbaik serta waktu pemesanan ulang paling lambat, jadi turunkan saat-saat itu sambil menjaga rasio kompresi yang baik merupakan tantangan bagi algoritma c-block.

Tabel 4 LATimes bit per gap dan running time (*k* = 200 dan pengkodean delta)

	c								
	70	100	150	200	300	400	500	1000	2000
Bit per celah	6.68	6.72	6.77	6.81	6.87	6.92	6.95	7.02	7.09
c-GreedyNN & kompres ulang	18 08	9 50	5 08	3 21	1 57	1 25	1 07	42	47

Tabel 5 bit FBIS per celah dan waktu berjalan (*k* = 200 dan pengkodean delta)

	c								
	70	100	150	200	300	400	500	1000	2000
Bit per celah	5.98	6.00	6.02	6.05	6.09	6.12	6.14	6.22	6.30
c-GreedyNN & kompres ulang	17 37	9 35	4 59	3 15	1 53	1 21	1 05	40	45

Metode ini meningkatkan rasio kompresi asli 7,25 (6,35) untuk LATimes (FBIS) koleksi dan pengkodean delta. Peningkatan ini lebih tinggi bila dibandingkan dengan secara acak rasio kompresi koleksi yang dipesan 7,65 (7,35). Pengorbanan antara kompresi akhir dicapai dan penggunaan waktu, dapat diparameterisasi dengan memilih nilai *c* dan *k* .

5. Strategi pengelompokan

5.1. Solusi pengelompokan untuk masalah penugasan ulang pengidentifikasi dokumen

Pekerjaan terbaru mengatasi masalah penugasan kembali pengidentifikasi dokumen dengan teknik pengelompokan. Ide utamanya adalah untuk meningkatkan lokalitas dalam koleksi dengan menugaskan dokumen serupa ke cluster yang sama, dan berharap bahwa kesenjangan d dapat diturunkan jika dokumen berada pada cluster yang sama. cluster diberi pengidentifikasi dekat. Untuk kasus khusus ini, ada dua kelompok algoritma dikembangkan untuk menghitung penugasan dokumen yang efisien: *penugasan top-down* dan *tugas dari bawah ke atas.* Skema penugasan top-down dimulai dengan seluruh koleksi dan secara rekursif membaginya menjadi sub-koleksi, memasukkan dokumen serupa ke dalam sub-koleksi yang sama. Setelah fase ini, algoritme menggabungkan sub-koleksi untuk mendapatkan satu dan kelompok dokumen terurut, yang digunakan untuk menghitung urutan penugasan. Dari bawah ke atas Skema dimulai dari sekumpulan dokumen, mengekstraksi rangkaian terpisah yang berisi dokumen serupa. Setiap urutan diurutkan, dan tugas akhir dihitung dengan mempertimbangkan a urutan sewenang-wenang antar urutan.

Algoritma top-down pertama diperkenalkan oleh Blandford dan Blelloch (2002), yang mengembangkan teknik yang meningkatkan rasio kompresi sekitar 14% dalam teks TREC koleksi. Teknik ini menggunakan grafik kesamaan seperti yang dijelaskan dalam Bagian 2.1, dan beroperasi dalam tiga fase berbeda. Tahap pertama membangun grafik kesamaan dokumen-dokumen dari file asli yang dibalik. Bagian kedua dari algoritma ini memanggil partisi grafik paket yang mengimplementasikan algoritma Metis (Karypis dan Kumar, 1995) untuk pemisahan secara rekursif grafik kesamaan yang dihasilkan oleh bagian pertama. Akhirnya, algoritma tersebut berlaku rotasi ke grafik berkerumun yang dihasilkan oleh bagian kedua untuk mengoptimalkan hasil yang diperoleh memesan. Tugas akhir pengidentifikasi dokumen diperoleh secara mendalam terlebih dahulu melintasi pohon pengelompokan yang dihasilkan. Sebagaimana dinyatakan dalam Blandford dan Blelloch (2002), membuat grafik kemiripan penuh adalah O(n2), jadi teknik mentahnya mungkin tidak cocok untuk itu

koleksi yang sangat besar. Namun demikian, efisiensi algoritma dapat dikendalikan oleh dua parameter: ÿ dan ÿ. Parameter pertama, ÿ , tindakan sebagai ambang batas untuk membuang suku-suku berfrekuensi tinggi, yaitu jika suatu suku *ti* mempunyai sejumlah kemunculan dokumen | *ti*| > ÿ dibuang untuk membuat grafik kemiripan. Sebenarnya algoritma ini bekerja dengan sampel grafik kemiripan penuh. Parameter ÿ menunjukkan seberapa agresif algoritme melakukan sub-sampel data: jika ukuran indeksnya *n,* maka ia akan mengekstrak satu elemen dari nÿ. Dengan menyetel ÿ dan ÿ, teknik ini dapat menyebabkan tradeoff antara efisiensi, waktu, dan penggunaan memori.

Pekerjaan di Silvestri dkk. (2004) mengusulkan pendekatan berbeda dengan *menugaskan* pengidentifikasi dokumen *dengan cepat* selama inversi kumpulan teks. Hal ini dilakukan dengan menghitung bentuk representasi *transaksional* dari dokumen, yang menyimpan untuk setiap dokumen *di* satu set bilangan bulat 4-byte yang mewakili MD5 Message-Digest (Rivest, 1992) dari setiap istilah yang muncul di *di* . Dengan menggunakan representasi ini, algoritma top-down baru yang disebut *membagi dua* disajikan dan dievaluasi, serta dua algoritma bottom-up: *k-means* dan *k-scan.* Teknik-teknik ini diuji dalam koleksi Kontes Pemrograman Google. Karena seluruh koleksi diurutkan saat pengindeksan, garis dasar yang dipilih untuk mengukur kinerja algoritme ini adalah kumpulan acak. Hasil kami sebelumnya dengan FBIS dan LATimes yang disajikan pada Bagian 3.3, menunjukkan bahwa pengurutan koleksi secara acak selalu menghasilkan rasio kompresi yang lebih buruk dibandingkan dengan *pengurutan alami,* yaitu meninggalkan koleksi dengan penomoran dokumen aslinya. Hal ini memotivasi kami untuk menerapkan kembali algoritme ini, namun dalam skenario penugasan ulang pengidentifikasi dokumen, dengan keberadaan file terbalik, daripada melakukan penugasan dengan cepat.

Dalam penerapan algoritme pengelompokan kami, konstruksi file terbalik tidak memerlukan hashing transaksional dari istilah tersebut. Untuk meningkatkan waktu komputasi kesamaan dokumen-ke-dokumen, kami bekerja tidak hanya dengan file terbalik, tetapi juga dengan file langsung. Struktur ini dapat dilihat sebagai file terbalik dimana kunci indeks adalah dokumennya, dan daftar posting berisi pengidentifikasi istilah. Perlu dicatat bahwa file langsung ini juga dikompresi dengan teknik yang digunakan untuk mengompresi file yang dibalik.

5.2. Membelah dua dan k-scan

Pada bagian ini, kami mengadaptasi algoritma membagi dua dan k-scan dari Silvestri et al. (2004) untuk skenario penugasan kembali. Algoritme pembagian dua termasuk dalam keluarga top-down, dan beroperasi sebagai berikut: pertama, algoritma ini secara acak memilih dua dokumen sebagai pusat massa dari dua partisi koleksi. Selanjutnya, ia menetapkan setiap dokumen dalam data masukan ke salah satu dari dua partisi, sesuai dengan kesamaan antara dokumen dan pusat massa. Teknik ini menjaga kedua partisi ini berukuran sama, sehingga banyak dokumen di akhir masukan mungkin masuk ke dalam cluster yang tidak diinginkan. Skema partisi ini disebut secara rekursif hingga partisi input hanya berisi satu elemen. Tahap penggabungan dilakukan dengan membandingkan batas kedua partisi. Jika kemiripan antara dokumen di tepi kanan cluster *D* dan dokumen di tepi kiri cluster *D* lebih kecil atau sama dengan kemiripan antara dokumen di tepi kanan cluster *D* dan dokumen di tepi kiri cluster *D* pada cluster pesanan akhir *D* harus mendahului *D* . Jika tidak, *D* harus mendahului D. Dalam kode semu, ÿ mewakili rangkaian dua set yang berisi pengidentifikasi dokumen. ,

K -scan adalah versi sederhana dari algoritma pengelompokan k-means yang populer. Algoritme mengurutkan elemen koleksi berdasarkan panjang dokumen menurun, dan memilih dokumen terbesar yang tidak dipilih sebagai perwakilan cluster. Kemudian ia menetapkan |D|/k ÿ 1 dokumen yang tidak dipilih yang paling mirip dengan perwakilan cluster. Ada sebuah

tatanan internal di setiap cluster, diberikan oleh ukuran kesamaan ini. Pemilihan tersebut perwakilan klaster dan penugasan dokumen ke klaster tersebut diulangi *k* waktu.

Pembagian Dua Algoritma (D)

```
1: Masukan:
       Kumpulan dokumen D
2: Keluaran:
       Daftar dengan pesanan akhir
3: jika |D| > 1 lalu
4: c1 ÿ dokumen acak(D);
5: c2 ÿ dokumen acak(D \ c1);
6:     D ÿ {c1};
7:     D ÿ {c2};
8:     untuk semua d ÿ D \ {c1, c2}
9:         jika (|D | ÿ ) maka ÿ (|D| ÿ |D| / 2)
10:            Tetapkan d ke partisi terkecil;
           kalau tidak
11:12:         jika kesamaan(c1, d) ÿkesamaan(c2, d) maka
13:            D ÿ D ÿ d;
14:        kalau tidak
15:            D ÿ D ÿ d
16:        berakhir jika
17:    berakhir jika
18:    akhir untuk
19:    D pesanan ÿ Membelah dua(D )
20:    D pesanan ÿ Membagi dua (D)
21:    jika sim(kanan(D ord),kiri(D ord))
                ÿ sim(kanan(D ord),kiri(D ord)) lalu
22:        Dord ÿ D pesanan pertama ÿ D
23:    kalau tidak
24:        Dord ÿ D pesanan pertama ÿ D
25:    berakhir jika
26:    kembali Dord
27: lain
28:    kembali D;
```

Pekerjaan di Silvestri dkk. (2004) menunjukkan bahwa ruang penyimpanan dan biaya komputasi dari algoritma pembagian dua keduanya superlineal O(|D| log(|D|)) dalam analisis asimtotik. Untuk k-scan, ruang yang ditempati adalah garis O(|S||D|) dan kompleksitasnya adalah O(|D|k), dimana |*S*| adalah panjang rata-rata dokumen.

5.3. Eksperimen dan hasil

Eksperimen dirancang untuk menilai kinerja teknik pengelompokan dan bandingkan dengan pendekatan berbasis TSP, untuk mengukur peningkatan kompresi rasio terhadap koleksi asli dan koleksi acak. Persamaan antara dokumen diukur dengan jarak Jaccard, mengulangi kondisi yang dijelaskan dalam Silvestri dkk. (2004). Untuk memanfaatkan transformasi SVD yang dihitung sebelumnya,

kami mengulangi eksperimen yang mengukur kemiripan antara dokumen dengan produk dalam dalam ruang dimensi tereduksi. Dalam kasus ini, parameter *k* dalam transformasi SVD ditetapkan menjadi 200, karena nilai ini terbukti layak (Bagian 3.3 dan 4).
Selanjutnya, parameter *k* yang muncul pada tabel mengacu pada jumlah scan *D* dalam algoritma k-scan. Dalam rangkaian percobaan ini, kompresi daftar posting dilakukan dengan pengkodean delta.

algoritma k-scan

1: **Masukan:**
Kumpulan *dokumenD*
Banyaknya barisan *k*

2: **Keluaran:** *k*
urutan terurut mewakili urutan akhir
3: *hasil* ÿ 4:
Urutkan *D* berdasarkan panjangnya; 5:
untuk *i* = 1 ... *k* **lakukan**
6: *pusat* ÿ pertama(D); **untuk**
7: **semua** *dj* ÿ *D* **lakukan**
8: sim[j] ÿ kesamaan(tengah, dj); 9: **akhir**
untuk
9: *M* ÿ 10: $\overline{Dk}$ ÿ 1 dokumen dengan nilai *sim* lebih tinggi;
hasil ÿ *hasil* ÿ *{pusat}* ÿ *M*
11: *D* ÿ *D* \ *(M* ÿ *{center})* 12: **akhir untuk**
13:
mengembalikan *hasil;*

Algoritma membagi dua mencapai keuntungan yang berbeda dalam bit per gap di setiap proses, karena pemilihan pusat acak untuk eksekusi yang berbeda, meskipun mereka stabil pada nilai tertentu. Dalam 10 kali menjalankan algoritma, untuk koleksi LATimes, nilai yang diperoleh berada dalam kisaran [6.82, 6.87] untuk jarak Jaccard dan [7.24, 7.32] untuk hasil kali dalam. Membandingkan nilai-nilai terburuk dalam rentang ini dengan urutan pengumpulan asli (acak) 7,25 (7,65), metode membagi dua memperoleh keuntungan dalam rasio kompresi sebesar 5,24% (10,19%) untuk Jarak Jaccard dan ÿ0,97% (4,30%) untuk Jarak Jaccard. produk dalam.

Hal utama yang perlu dipertimbangkan adalah bahwa keuntungan yang diperoleh sehubungan dengan tatanan alam sangatlah kecil, atau mungkin tidak ada sama sekali. Pengamatan ini dibuktikan dengan hasil algoritma pembagian dua dalam koleksi FBIS, yang kami lewati di sini. Komentar mengenai perbedaan antara hasil yang diperoleh dengan dua ukuran kemiripan yang berbeda untuk setiap percobaan dapat ditemukan di akhir bagian ini.

Tabel 6(7) menunjukkan hasil kompresi akhir yang diukur dalam bit per gap untuk koleksi LATimes (FBIS) untuk algoritma k-scan. Baris pertama menyajikan data yang diperoleh dengan mengukur kemiripan dengan hasil kali dalam dan 200 dimensi dalam ruang tereduksi, yang kondisinya persis sama dengan eksperimen dengan blok-c (Bagian 4) dan algoritma pembagian dua. Baris kedua menunjukkan hasil dengan jarak Jaccard. Parameter *k* adalah singkatan dari jumlah pemindaian algoritma. Secara keseluruhan hasilnya memberikan nilai kompresi yang baik untuk nilai *k* yang cukup besar (1000). Namun, mengingat variasi k, ada titik di mana celah bit per dokumen berhenti berkurang, dan sedikit meningkat hingga mencapai nilai batas di mana *k* = |D|, yang hanya mengurutkan koleksi berdasarkan panjang dokumen.

Tabel 6 k-scan LATimes bit per gap

	k						
	100 200 300 400	500 1000 2000 3000 30000 ⋯ 131896					
Produk dalam 6,79 6,72 6,70 6,67 6,66 6,62			6.60	6.60 6.70			⋯7.65
Jaccard 6,82 6,74 6,70 6,68 6,67 6,64			6.63	6.62	6.71		⋯7.65

Tabel 7 k-scan bit FBIS per celah

	k						
	100 200 300 400	500 1000 2000 3000 30000 ⋯ 130471					
Produk dalam 6,46 6,39 6,36 6,34 6,33 6,29			6.23	6.21	6.26		⋯7.35
Jaccard 6,54 6,47 6,44 6,41 6,39 6,29			6.23	6.22 6.27			⋯7.35

Tabel 8 k-scan + TSP LATimes bit per gap (pengkodean delta)

	k				
	100	200	300	400	500
Produk dalam di bawah 200 dimensi 6,34 6,35 6,36 6,37 6,37					
Jaccard 6.25 6.28 6.29 6.30 6.30					

Tabel 9 k-scan + TSP FBIS bit per gap (pengkodean delta)

	k				
	100	200	300	400	500
Produk dalam di bawah 200 dimensi 5,93 5,94 5,95 5,96 5,97					
Jaccard 5,84 5,87 5,87 5,89 5,89					

Sungguh luar biasa bahwa nilai terbaik diperoleh dengan algoritma k-scan di LATimes koleksi (6.60) dan nilai yang diberikan oleh algoritma blok-c serupa (6.68). Ini bukan berlaku untuk koleksi FBIS, di mana algoritma c-block (5.98) memperoleh nilai yang lebih baik daripada k -scan (6.23). Dalam hal penguatan terhadap rasio kompresi asli (6,35) ini akan menjadi 1,9% vs 5,9%. Perbandingan ini adil, karena metrik yang digunakan oleh kedua algoritma pada ukuran ini sama persis (hasil kali dalam pada ruang $k = 200$).

5.4. Kombinasi TSP dan pengelompokan

Terakhir, kami menyajikan teknik yang menggabungkan pendekatan terbaik sebelumnya: yang nyata kinerja dalam bit per celah TSP dan efisiensi konsumsi waktu pengelompokan. Ide dasarnya sama seperti pada algoritma c-block (lihat Bagian 4), yaitu split the pengumpulan menjadi beberapa bagian dan menjalankan TSP di setiap bagian, sehingga waktu berjalan keseluruhan diturunkan dari beberapa jam hingga hanya beberapa menit. Perbedaannya terletak pada cara pemilihan balok; alih-alih untuk menggerakkan partisi *mentah* dari blok berukuran sama, kami memilih setiap subkoleksi dengan algoritma *k*- scan. Dengan cara ini, diharapkan setiap partisi menyimpan dokumen serupa, sehingga TSP akan meningkatkan hasil yang sudah baik.

Tabel 8 dan 9 menunjukkan hasil yang diperoleh dari pengumpulan LATimes dan FBIS untuk ini teknik, masing-masing. Kinerja algoritmanya kurang lebih sama dengan

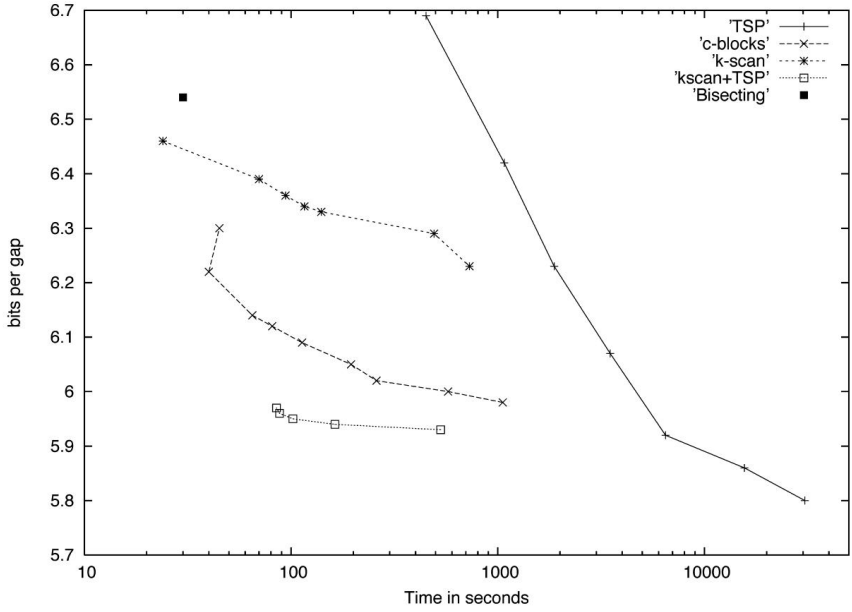
menjalankan Greedy-NN (TSP untuk seluruh koleksi) dengan pengurangan ruang 200 dimensi (hasil terbaik yang diperoleh sejauh ini). Satu hal yang perlu dipertimbangkan adalah bahwa teknik ini memperoleh hasil yang lebih baik daripada blok-c di kedua koleksi. Hal ini membuktikan bahwa teknik berbasis TSP bekerja dengan baik ketika pengumpulan membawa *tatanan alami* dengan rasio kompresi yang baik, yang lebih sulit ditingkatkan dengan algoritma apa pun, namun lebih sulit lagi ditingkatkan dengan pendekatan pengelompokan yang telah diuji sebelumnya. Selain itu, waktu penyusunan ulang untuk nilai *k* yang ditampilkan berada dalam urutan blok c, karena ukuran partisinya seimbang. Tentu saja, hal ini juga berlaku untuk algoritme yang dijelaskan di bagian sebelumnya, meskipun teknik khusus ini melakukan penyusunan ulang setiap subkoleksi selain menghitung cluster.

Dapat diketahui bahwa pengukuran Jaccard memiliki kinerja yang lebih baik daripada hasil kali dalam pada algoritma pembagian dua dan k-scan + TSP, meskipun hasil kali dalam berperilaku sedikit lebih baik untuk k-scan, terutama dengan nilai k yang rendah. Masalah dengan jarak Jaccard adalah bahwa jarak tersebut merupakan ukuran yang tidak terdefinisi dalam ruang tereduksi SVD, karena berhubungan dengan perpotongan suku aslinya. Namun, hasil produk dalam berguna untuk membandingkan secara adil teknik pengelompokan dan blok-c.

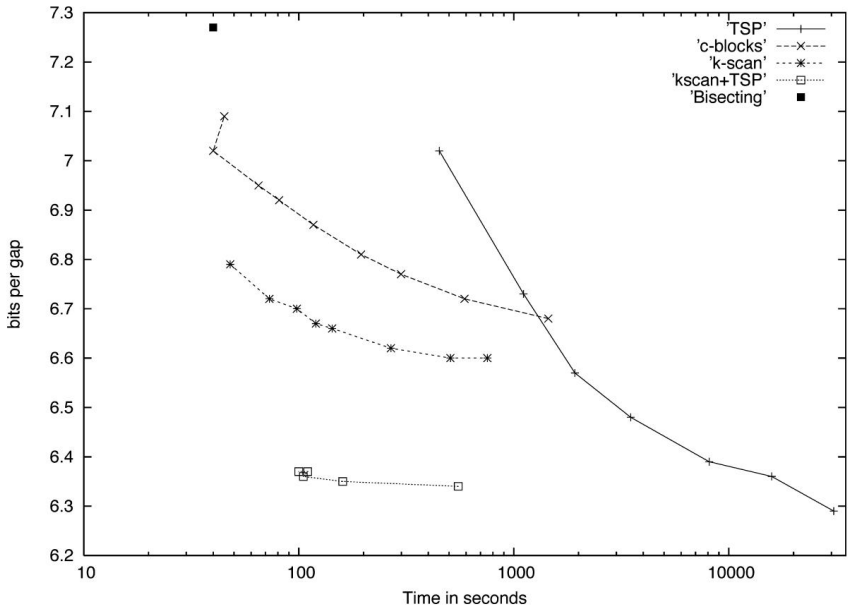
6. Diskusi dan pekerjaan masa depan

Pekerjaan yang disajikan di sini menunjukkan secara empiris bahwa heuristik TSP adalah strategi penataan ulang yang baik. Hal ini didukung oleh implementasi, pengujian dan perbandingan dengan algoritma clustering dan khususnya dengan kombinasi dua teknik utama. Sebagai ringkasan hasil yang disajikan di seluruh makalah, Gambar. Gambar 2 dan 3 menunjukkan beberapa hasil yang diperoleh dari algoritma yang dijelaskan di sini. Plot tersebut memberikan beberapa rincian tentang efektivitas dan efisiensi trade-off yang dapat dicapai dengan kombinasi TSP dan berbasis klaster. Sumbu x menunjukkan

menyusun ulang dan mengompresi ulang waktu dalam skala logaritmik, dan sumbu y mewakili bit



Gambar 2 Ringkasan hasil koleksi LATimes



Gambar 3 Ringkasan hasil pengumpulan FBIS

per kesenjangan yang dicapai untuk pengkodean delta. Setiap baris dalam plot merupakan singkatan dari teknik penataan ulang yang berbeda-beda, yaitu TSP (Bagian 2), c-block (Bagian 4), k-scan (Bagian 5.2), membagi dua (Bagian 5.2) dan kombinasi TSP dan pengelompokan menggunakan k-scan (Bagian 5.4).

Misalnya, dimungkinkan untuk mencapai pengurangan rasio kompresi sebesar 13,24% (6,29 bit per gap) (LATimes, delta coding) dengan biaya waktu lebih dari 8 jam mengemudikan algoritma TSP saja, sedangkan TSP +k-scan kombinasi memberikan rasio 12,55% (6,34 bit per gap) dengan upaya komputasi yang jauh lebih sedikit (dalam urutan 10 menit).

Terlepas dari kenyataan bahwa teknik berbasis TSP berkinerja lebih baik dibandingkan pendekatan lain dan dapat disesuaikan dengan situasi lain agar sesuai dengan konsumsi waktu dan penggunaan memori (pengurangan SVD, partisi koleksi, dan kombinasi dengan pengelompokan) hal ini tidak serta merta memberikan hasil yang optimal. solusi untuk masalah penugasan kembali pengidentifikasi dokumen.

Untuk menemukan solusi yang lebih baik daripada yang diberikan oleh strategi TSP, perlu ditemukan heuristik yang memperhitungkan fungsi biaya yang tepat untuk diminimalkan. Minimisasi ini harus memungkinkan adanya skema pengkodean, karena tujuannya adalah untuk meminimalkan fungsi

$$\text{biaya}(\ddot{y}) = \sum_{t=1}^{T} \text{panjang}(\ddot{y}(dt,1)) + \sum_{saya=1}^{f\,t} \text{panjang}(\ddot{y}(dt,i+1)\ \ddot{y}\ \ddot{y}(dt,i)) \qquad (9)$$

dimana ÿ adalah fungsi bijektif penugasan ulang yang memetakan setiap pengidentifikasi dokumen dalam rentang {1 ... d} ke yang baru dalam interval yang sama, dan panjang(x) memberikan jumlah total bit yang diperlukan untuk mengkodekan bilangan bulat x. Kebanyakan kode statis mewakili bilangan bulat *x* dengan bit O(log(x)) . Untuk mendapatkan rasio kompresi yang baik, kita harus meminimalkan produk d-gap. Dengan cara ini, log produk dan jumlah log akan menjadi minimal, dengan konsekuensi praktis dalam daftar postingan berkode akhir. Dalam konteks ini, mengingat masalah ini, TSP, meskipun memberikan hasil yang baik, hanyalah sebuah strategi untuk mengatasi pengidentifikasi dokumen

masalah penugasan kembali. Namun, ada kemungkinan penjelasan mengapa ia bekerja dengan baik dalam mengurangi log d-gap.

Misalkan matriks jarak Ld×d dimana setiap *li j* mengukur jarak antara dokumen *i* dan j. Mengingat dokumen sebagai himpunan suku, jarak dapat diukur sebagai |*(di* ÿ *dj)* \ *(di* ÿ dj)|. Traversal yang ditemukan oleh solusi eksak TSP meminimalkan fungsi *sumdist(ÿ)* = i=1 lÿ(di)ÿ(di+1). Bayangkan file terbalik sebagai matriks biner B berukuran *t* × *d* , di mana posisi *bi j* mewakili kemunculan istilah *i* dalam dokumen j. Untuk setiap baris matriks biner ini, setiap kali saklar 0–1 atau 1–0 muncul, *jumlah saklar (ÿ)* bertambah satu, sehingga urutan yang meminimalkan *jumlah saklar tersebut (ÿ)* adalah urutan yang meminimalkan jumlah saklar tersebut. Hal ini akan memaksimalkan perpotongan bitwise vektor dokumen, namun tidak berarti meminimalkan rata-rata jumlah logaritma d-gap. Namun, hal ini memastikan bahwa konfigurasi akhir dari matriks biner memiliki sejumlah besar angka 1 berturut-turut di setiap baris, sehingga masuk akal untuk mengharapkan jumlah d-gap logaritma rata-rata yang rendah (karena logaritma tidak memberikan penalti yang lebih besar terhadap d-gap yang sama. karena mendapat manfaat dari yang lebih kecil).

Dÿ1

Hal ini dimungkinkan untuk menelusuri jalur penelitian aktif baru, menghasilkan heuristik baru berdasarkan fungsi biaya optimal dan/atau karakterisasi jarak yang dicapai solusi tertentu sehubungan dengan optimal. Hal ini juga berlaku untuk solusi TSP: selama algoritme dalam ruang dimensi tereduksi ini bekerja dengan baik, kita dapat melakukan karakterisasi formal hingga jarak solusi optimal yang dicapai, dengan solusi heuristik semacam ini.

Kombinasi pengelompokan dan TSP juga akan menghadapi masalah skalabilitas yang sama seperti pendekatan khusus TSP yang dijelaskan dalam Shieh dkk. (2003) dalam koleksi yang sangat besar. Kombinasi TSP, SVD dan pengelompokan mungkin cocok untuk kasus-kasus ini, dan akan menarik untuk mengukur keuntungan yang dapat dicapai oleh teknik-teknik ini, karena pengelompokan memberikan hasil yang baik tetapi hasilnya menjadi lebih tinggi dengan menerapkan TSP di setiap klaster. Selain itu, hal ini terutama diindikasikan untuk teknik partisi kumpulan yang dapat menghasilkan cluster yang tidak seimbang, seperti algoritma *k-means* (Jain et al., 1999). Efek negatif dari cluster besar mungkin akan dikurangi dengan SVD, sehingga teknik TSP dapat dijalankan dengan waktu yang dapat diterima.

Masalah skalabilitas lainnya adalah efisiensi langkah SVD. Seperti yang telah kita lihat, pengurangan dimensi memungkinkan implementasi yang efisien dari pendekatan berbasis TSP, namun ketika berhadapan dengan skalabilitas web, ketika melibatkan beberapa Gigabyte atau bahkan Terabyte, implementasi SVD yang lebih efisien menjadi perlu, lihat untuk contoh Kokiopoulou dan Saad (2004), atau transformasi matriks lain yang berbeda, dengan sifat serupa dengan SVD, namun lebih efisien dalam waktu komputasi.

7. Kesimpulan

Kami menyajikan perkiraan cerdas untuk masalah penugasan ulang pengidentifikasi dokumen dengan menggunakan pengurangan dimensi sebelumnya dengan SVD. Hasil yang disajikan memungkinkan untuk melaporkan metode hemat waktu yang menghasilkan keuntungan pengurangan file terbalik yang baik. Secara konkret, kami menerapkan pendekatan TSP Greedy-NN dalam ruang dimensi tereduksi dan satu varian, yang menerapkan solusi ini pada sub-kumpulan data asli, dan kemudian menyusun ulang data tersebut. Kami juga mempresentasikan implementasi dua teknik pengelompokan untuk masalah penugasan ulang dan mengevaluasi serta membandingkannya dengan penataan ulang berbasis TSP. Terakhir, di bagian diskusi kami menunjukkan perbandingan berbagai teknik dan memberikan wawasan mengenai aspek teoretis dan masalah skalabilitas dari karya yang diusulkan, serta jalur penelitian di masa depan.

Ucapan Terima Kasih Pekerjaan yang dilaporkan di sini didanai bersama oleh “Ministerio de Educaci´on y Ciencia” dan dana FEDER di bawah proyek penelitian TIC2002-00947 dan TIN2005-08521-C02-02 dan “Xunta de Galicia” di bawah proyek PGIDT03PXIC10501PN.

Referensi

Bartell, BT, Cottrel, GW, & Belew, RK (1992). Pengindeksan semantik laten adalah kasus khusus penskalaan multidimensi yang optimal. Dalam *Prosiding Konferensi Internasional ACM SIGIR Tahunan ke-15 tentang Penelitian dan Pengembangan Pencarian Informasi* (hlm. 161–167).

Blanco, R., & Barreiro, A. (2005). Penugasan ulang pengidentifikasi dokumen melalui pengurangan dimensi. Dalam *Prosiding Konferensi Eropa ke-27 tentang Penelitian IR, ECIR 2005,* LNCS 3408 (hlm. 375–387).

Blandford, D., & Blelloch, G. (2002). Kompresi indeks melalui penataan ulang dokumen. Dalam: *Prosiding Konferensi Kompresi Data IEEE (DCC'02),* hlm.342–351.

Deerwester, S., Dumais, ST, Furnas, GW, Landauer, TK, & Harshman, R. (1990). Pengindeksan dengan analisis semantik laten. *Jurnal Masyarakat Amerika untuk Ilmu Informasi,* 41(6), 391–407.

Dumais, ST (1994). Pengindeksan Semantik Laten (LSI): Laporan TREC-3. Dalam *Publikasi Khusus NIST 500-225: Prosiding Konferensi Pengambilan Teks Ketiga (TREC-3),* November 1994.

Elias, P. (1975). Kumpulan kata kode universal dan representasi bilangan bulat. *Transaksi IEEE pada Teori Informasi,* 21(2), 194–203.

Gallager, R., & van Voorhis, D. (1975). Kode sumber optimal untuk alfabet bilangan bulat yang terdistribusi secara geometris. *Transaksi IEEE pada Teori Informasi, 21,* 228–230.

Golomb, S. (1966). Pengkodean jangka waktu. *Transaksi IEEE pada Teori Informasi, 12,* 399–401. http://

mg4j.dsi.unimi.it/ MG4J (Mengelola Gigabyte untuk Java). http://tedlab.mit.edu/ÿdr/

SVDLIBC/ SVDLIBC. http://www.cs.mu.oz.au.mg/ Mengelola Gigabyte.

Karypis, G., & Kumar, V. (1995). *Skema Multilevel yang Cepat dan Berkualitas Tinggi untuk Mematisi Grafik Tidak Beraturan* (Tech. Rep. No. TR 95-035). Departemen Ilmu Komputer, Universitas Minnesota, Jain, AK, Murty, MN,

& Flynn, PJ (1999). Pengelompokan data: Sebuah tinjauan. *Survei Komputasi ACM,* 31(3), 264–323.

Kokiopoulou, E., & Saad, Y. (2004). Pemfilteran polinomial dalam pengindeksan semantik laten untuk pengambilan informasi. Dalam *Prosiding Konferensi Internasional Tahunan ACM SIGIR ke-27 tentang Penelitian dan Pengembangan Pencarian Informasi,* hal. 104–111.

Moffat, A., & Turpin, A. (2002). *Algoritma kompresi dan pengkodean,* Kluwer.

Moffat, A., & Stuiver, L. (2000). Pengkodean interpolatif biner untuk kompresi indeks yang efektif. *Pengambilan Informasi,* 3(1), 25–47.

Rivest, R. (1992). RFC 1321: Algoritma md5.

Shieh, W.-Y., Chen, T.-F., Shann, JJ-J., & Chung, C.-P. (2003). Kompresi file terbalik melalui penugasan kembali pengidentifikasi dokumen. Dalam *Pemrosesan dan Manajemen Informasi,* 39(1), 117–131.

Silvestri, F., Orlando, S., & Perego, R. (2004). Menetapkan pengidentifikasi ke dokumen untuk meningkatkan properti pengelompokan indeks teks lengkap. Dalam *Prosiding Konferensi Internasional ACM SIGIR Tahunan ke-27 tentang Penelitian dan Pengembangan Pencarian Informasi,* hal. 305–312.

Witten, IH, Moffat, A., & Bell, TC (1999). *Mengelola gigabyte—Mengompresi dan mengindeks dokumen dan gambar* (edisi ke-2). San Francisco: Penerbitan Morgan Kaufmann.