ALGORITME PELACAKAN KATA DENGAN TOLERANSI KESALAHAN

L. Yohanes Stefanus

Fakultas Ilmu Komputer, Universitas Indonesia

Makalah ini membahas salah satu masalah yang sangat umum dalam pemrosesan teks dengan bantuan komputer, yaitu bagaimana melacak dengan cepat suatu kata atau ungkapan tertentu dalam suatu berkas teks yang besar. Dalam banyak situasi, kata yang akan dilacak itu mungkin tidak diketahui dengan persis. Misalnya, ejaan kata itu sendiri kurang betul, atau teks tersebut mengandung kesalahan-kesalahan penulisan. Jadi bagaimana seharusnya kriteria untuk menentukan kesuksesan pelacakan? Jika sistem disuruh mencari kata "objek" dalam teks yang mengandung kata "obyek", apakah sistem harus menyatakan bahwa pelacakan itu berhasil menemukan kata yang sesuai?

Untuk membantu manusia yang pada umumnya bersifat mudah membuat kesalahan, sistem pelacakan teks sebaiknya bekerja berdasarkan kriteria kesuksesan pelacakan yang longgar. Tentu saja kelonggaran ini harus dibatasi. Kita tidak menginginkan sistem yang melemparkan ratusan jawaban untuk satu permintaan melacak kata yang kita yakin ejaannya benar. Sebaiknya sistem itu bekerja sebagai berikut: mula-mula ia akan berusaha menemukan kata yang persis; kalau kata yang persis tidak ada, ia akan berusaha menemukan kata yang ejaannya paling mendekati menurut ukuran kedekatan tertentu; dan seterusnya. Yang menjadi persoalan adalah bagaimana bentuk ukuran kedekatan tersebut.

Ukuran kedekatan yang sering dipakai adalah **jarak edit** yang dikemukakan oleh Levenshtein [2]. Suatu kata *A* dikatakan berjarak edit *k* ke kata *B* jika kita dapat mentransformasikan *A* menjadi *B* dengan sederetan *k* operasi yang berupa: penyisipan satu huruf pada *A*, atau penghapusan satu huruf dari *A*, atau substitusi satu huruf pada *A*.
Sebagai contoh,

- jarak edit antara "teks" dan "tak" adalah 2, karena untuk mentransformasikan "teks" menjadi "tak" diperlukan satu operasi substitusi (huruf e digantikan oleh huruf a) dan satu operasi penghapusan (huruf s dihapus).
- jarak edit antara "obyek" dan "objek" adalah 1, karena "obyek" dapat ditransformasikan menjadi "objek" dengan satu operasi substitusi.

Dalam praktek, kita perlu menentukan jarak edit maksimum yang layak, misalnya 3. Jarak edit maksimum yang terpilih sepatutnya didukung oleh suatu penelitian lapangan. Dengan kata lain, jarak edit maksimum itu harus sesuai dengan kebutuhan para pemakai. Di samping itu, mungkin kita perlu pula membatasi jenis operasi edit atau memberi bobot yang berbeda kepada masing-masing operasi edit itu.

Salah satu algoritme pelacakan teks yang cukup cepat dan sederhana adalah algoritme yang dikemukakan oleh Wu dan Manber [3]. Algoritme Wu-Manber ini sanggup melacak teks dengan jarak edit yang fleksibel. Algoritme Wu-Manber dibangun dari algoritme pelacakan teks yang diusulkan oleh Baeza-Yates dan Gonnet [1].

Algoritme Wu-Manber

Untuk mempermudah penjelasan, kita mulai dengan deskripsi untuk kasus dengan jarak edit maksimum nol, yaitu kasus yang menuntut pemadanan secara persis. Kemudian kita meninjau kasus dengan jarak edit maksimum lebih dari nol.

Andaikan kita ingin melacak kata $P = p_1p_2...p_m$ dalam suatu berkas teks besar $T = t_1t_2...t_n$. Andaikan R suatu array bit berukuran m, dengan m menyatakan panjang kata yang ingin dilacak. Kita menyatakan nilai dari array R setelah karakter ke j dari teks diproses, dengan notasi R_j. Array R_j mengandung informasi tentang semua kepadanan dari prefiks-prefiks P yang berakhir pada karakter ke j. $R_j[i] = 1$ jika i karakter pertama dari P berpadanan persis dengan i karakter terakhir sampai dengan karakter ke j dalam teks T (yaitu $p_1p_2...p_i = t_{j-i+1}t_{j-i+2}...t_j$). Ketika kita membaca karakter ke $j+1$ dari teks, yaitu t_{j+1}, kita perlu menentukan apakah

karakter tersebut dapat meneruskan kepadanan-kepadanan parsial yang telah diperoleh sejauh ini. Untuk setiap i yang sedemikian sehingga $R_j[i] = 1$ kita perlu memeriksa apakah t_{j+1} sama dengan p_{i+1}. Jika $R_j[i] = 0$, maka tidak ada kepadanan sampai dengan karakter ke i dan tentu saja tidak akan ada kepadanan sampai dengan karakter ke $i+1$. Jika $t_{j+1} = p_1$ maka $R_{j+1}[1] = 1$. Jika $R_{j+1}[m] = 1$ maka kita mendapatkan kepadanan yang lengkap mulai dari karakter ke $j-m+2$ sampai dengan karakter ke $j+1$; dengan kata lain, kita menemukan kata P dalam teks T.

Uraian di atas hanya mengemukakan ide pokok dari algoritme pelacakan. Implementasinya pada komputer dapat dibuat lebih efisien dari pada yang diuraikan di atas. Lihat [1] dan [3] untuk mendapatkan perincian teknisnya.

Sekarang kita tinjau kasus dengan satu kesalahan ejaan (jarak edit maksimum 1). Andaikan hanya satu operasi penyisipan boleh dilakukan pada teks P. Dengan kata lain, kita ingin mendapatkan semua interval berukuran $m+1$ atau kurang dalam teks T yang mengandung kata P sebagai subbarisan. Kita definisikan R seperti dulu, tetapi kita sekarang mempunyai dua kemungkinan untuk tiap kepadanan prefiks dari P terhadap T. Kemungkinan pertama, kita mendapatkan suatu kepadanan persis. Kemungkinan kedua, kita mendapatkan suatu kepadanan dengan satu operasi penyisipan. Dengan demikian, kita perlu mempergunakan array lain yang akan kita beri notasi R_j^1. Array ini akan mengandung informasi tentang semua kepadanan sampai dengan karakter ke j dengan paling banyak satu penyisipan. $R_j^1[i] = 1$ jika i karakter pertama dari P berpadanan dengan i dari $i+1$ karakter terakhir sampai dengan karakter ke j dalam teks T. Jika kita dapat menyediakan baik R maupun R^1, maka kita bisa mendapatkan semua kepadanan dengan paling banyak satu operasi penyisipan. Dalam hal ini, $R_j[m] = 1$ menyatakan bahwa terdapat suatu kepadanan persis dan $R_j^1[m] = 1$ menyatakan bahwa terdapat suatu kepadanan dengan paling banyak satu penyisipan.

Cara untuk memperbarui array R_j menjadi array R_{j+1} adalah seperti dulu. Cara untuk memperbarui array R^1 adalah sebagai berikut. Ada dua kasus yang harus kita perhatikan untuk suatu kepadanan dengan paling banyak satu penyisipan pada waktu kita sampai pada karakter t_{j+1}:

(1) Terdapat kepadanan persis dari i karakter pertama dari P dengan bagian dari teks T sampai dengan t_j. Dalam kasus ini, penyisipan t_{j+1} pada ujung P membuahkan suatu kepadanan dengan satu penyisipan.
(2) Terdapat kepadanan dari i-1 karakter pertama dari P dengan bagian dari teks T sampai dengan t_j dengan satu penyisipan dan $t_{j+1} = p_i$. Dalam kasus ini, penyisipan tidak pada ujung P.

Selanjutnya kita tinjau kasus kepadanan dengan satu operasi penghapusan. Kita definisikan R dan R^1 seperti tadi, kecuali penyisipan digantikan oleh penghapusan. Di sini juga ada dua kasus yang harus kita perhatikan untuk suatu kepadanan dengan paling banyak satu penghapusan pada waktu kita membaca karakter t_{j+1}:

(1) Terdapat kepadanan persis dari i-1 karakter pertama dari P dengan bagian dari teks T sampai dengan karakter t_{j+1}. Kasus ini bersesuaian dengan penghapusan p_i dan kepadanan terhadap i-1 karakter pertama dari P.
(2) Terdapat kepadanan dari i-1 karakter pertama dari P dengan bagian dari teks T sampai dengan karakter t_j dengan satu penghapusan dan $t_{j+1} = p_i$. Dalam kasus ini, penghapusan tidak pada ujung P.

Sekarang kita tinjau kasus dengan satu substitusi, yaitu kita membolehkan penggantian satu karakter dari P oleh satu karakter dari T. Lagi kita menghadapi dua kasus:

(1) Terdapat kepadanan persis dari i-1 karakter pertama dari P dengan bagian dari teks T sampai dengan karakter t_j. Kasus ini bersesuaian dengan penggantian t_{j+1} oleh p_i dan pemadanan terhadap i-1 karakter pertama dari P.

(2) Terdapat kepadanan dari *i-1* karakter sampai dengan karakter t_j dengan satu substitusi dan $t_{j+1} = p_i$. Dalam kasus ini substitusi tidak pada ujung dari *P*.

Dengan mengkombinasikan ketiga kasus di atas: kasus dengan satu penyisipan, kasus dengan satu penghapusan, dan kasus dengan satu substitusi, kita dapat merumuskan kasus umum dengan jumlah kesalahan sampai dengan *k* (jarak edit maksimum *k*). Yang dimaksudkan dengan satu kesalahan di sini bisa berupa penyisipan, penghapusan, atau substitusi. Secara umum, kita harus menyediakan array-array R, R^1, ... R^k sedemikian sehingga array R^d menyimpan informasi tentang semua kepadanan dengan edit maksimum *d*. Dalam memperbarui array R_j^d menjadi array R_{j+1}^d, kita perlu memperhatikan 4 kemungkinan terjadinya kepadanan dari *i* karakter pertama dari *P* with jarak edit maksimum *d* dengan bagian dari teks *T* sampai dengan karakter t_{j+1}:

(1) Terdapat kepadanan dari *i-1* karakter pertama dari *P* dengan bagian dari teks *T* sampai dengan karakter t_j dengan jarak edit maksimum *d*, dan $t_{j+1} = p_i$. Kasus ini bersesuaian dengan pemadanan t_{j+1}.
(2) Terdapat kepadanan dari *i-1* karakter pertama dari *P* dengan bagian dari teks *T* sampai dengan karakter t_j dengan jarak edit maksimum *d-1*. Kasus ini bersesuaian dengan substitusi t_{j+1}.
(3) Terdapat kepadanan dari *i-1* karakter pertama dari *P* dengan bagian dari teks *T* sampai dengan karakter t_{j+1} dengan jarak edit maksimum *d-1*. Kasus ini bersesuaian dengan penghapusan p_i
(4) Terdapat kepadanan dari *i* karakter pertama dari *P* dengan bagian dari teks *T* sampai dengan karakter t_j dengan jarak edit maksimum *d-1*. Kasus ini bersesuaian dengan penyisipan t_{j+1}.

Penutup dan Saran:

Dengan tersedianya algoritme Wu-Manber itu, pelacakan teks dengan toleransi kesalahan tertentu tidak perlu lagi dilakukan secara manual, tetapi dapat dilakukan dengan bantuan komputer. Pelacakan teks secara elektronis ini tidak saja lebih cepat tetapi juga lebih handal. Manusia memang secara hakekat tidak cocok untuk pekerjaan yang terus-menerus

rumit dan membosankan. Untuk mengatasi kelemahan manusia inilah komputer selayaknya dikerahkan.

Masih diperlukan penelitian untuk mengetahui jarak edit maksimum yang cocok untuk teks berbahasa Indonesia. Penelitian-penelitian semacam ini memerlukan data yang ekstensif dan representatif. Alangkah baiknya ada suatu kerjasama antara pakar bahasa Indonesia dan pakar ilmu komputer untuk membangun suatu kumpulan teks yang ekstensif dan representatif yang dapat digunakan dalam berbagai penelitian berbantuan teknologi komputer yang berkenaan dengan teks berbahasa Indonesia. Koleksi teks ini bisa diambil dari surat kabar, dokumen pemerintah, dokumen badan usaha, majalah ilmiah, majalah populer, dan buku-buku pelajaran.

Daftar Pustaka:

[1] Baeza-Yates, R. dan Gonnet, G.H. A new approach to text searching. *Comm. ACM 35, 10* (Oct. 1992), 74-82.

[2] Levenshtein, V. I. Binary codes capable of correcting deletions, insertions, and reversals. *Sov. Phys. Dokl.* (Feb. 1966), 707-710.

[3] Wu, S. dan Manber, U. Fast text searching allowing errors. *Comm. ACM 35, 10* (Oct. 1992), 83-91.