Jurnal Ilmiah Ilmu Komputer, Vol 15 No. 2, Desember 2010 : 25 - 29

Koreksi Ejaan Query Bahasa Indonesia Menggunakan Algoritme Damerau Levenshtein

Utis Sutisna, Julio Adisantoso

Departemen Ilmu Komputer, Institut Pertanian Bogor, Jl. Meranti Wing 20 Lv.V, Bogor, Jawa Barat, 16680

Abstract---Query spelling on search engine is important to improve the quality information searching result. When user types query for search engine input, sometime spelling mistakes occurred due to position of keyboard and finger movement while typing. As an effect, searching result is incorrect and when user misspells the query, information obtained will not succeed. Therefore, search engine requires an application of spelling corrections. This research proposes correction process for query spelling by giving words suggestions which are obtained by calculating edit distance for each corrected word towards every word in dictionary. The concept for calculating edit distance uses Damerau Levenshtein algorithm which consists of 4 operations: (1) insertion, (2) substitution, (3) deletion, and (4) transposition. This research shows that implementation of Damerau Levenshtein algorithm is able to increase recall-precision value in information retrieval system. It is shown by the increasing of average recall-precision value at 44,82% after correction.

Keywords : Damerau Levenshtein, Algoritme Damerau Levenshtein Metric, edit distance

PENDAHULUAN

Kata kunci atau yang biasa disebut dengan *query* pada pencarian informasi dari sebuah *search engine* digunakan sebagai kriteria pencarian yang tepat dan sesuai dengan kebutuhan. Ejaan kata kunci yang benar menjadi penting untuk meningkatkan hasil pencarian informasi. Ketika pengguna menulis *query* sebagai masukan pada sistem pencari, muncul kesalahan ejaan disebabkan posisi tombol papan ketik dan pergerakan jari sehingga hasil pencarian bersifat salah. Oleh karena itu, diperlukan suatu aplikasi yang dapat mengoreksi kesalahan ejaan. Pengoreksian ejaan ini dapat dilakukan dengan memberikan ejaan kata yang benar yaitu dengan memberikan usulan ejaan kata yang mirip berdasarkan kamus.

Penelitian tentang pengoreksian ejaan bahasa Indonesia juga pernah dilakukan oleh Primasari (1997), melakukan penelitian tentang pencarian dan temu kembali nama berdasarkan kesamaan fonetik. Arumsari (1998) dengan menggunakan metode jarak *edit*. Wahyudin (1999) dengan menggunakan algoritme trigram untuk mendapatkan kata-kata perkiraan dari kata yang dinyatakan salah eja. Arumsari (1998) menentukan jarak *edit* diantara dua *string* dari operasi yang dilakukan yaitu: (1) operasi penyisipan (*insertion*), (2) operasi penghapusan (*deletion*), dan (3) operasi penggantian (*subtitution*) sebuah huruf. Pada penelitian ini, pengoreksian ejaan akan dilakukan dengan algoritme Damerau Levenshtein dengan metode jarak *edit*. Operasi yang dilakukan tidak hanya tiga operasi seperti yang dilakukan dalam penelitian Arumsari (1998). Tetapi, juga diperhatikan operasi penukaran (*transposition*) posisi sebuah huruf yang berdekatan. Sehingga perolehan kata ejaan yang benar lebih optimal.

METODOLOGI

Menurut Damerau dalam Wahyudin (1999) menyimpulkan 80% kesalahan ejaan dapat disebabkan karena empat hal, yaitu:

1. Penggantian satu huruf
2. Penyisipan satu huruf
3. Penghilangan satu huruf
4. Penukaran dua huruf berdekatan.

Kesalahan ejaan juga dapat disebabkan oleh beberapa hal, diantaranya:

1. Ketidaktahuan penulisan. Kesalahan ini biasanya konsisten dan kemungkinan berhubungan dengan bunyi kata dan penulisan yang seharusnya.
2. Kesalahan dalam pengetikan yang lebih tidak konsisten tapi mungkin berhubungan erat dengan posisi tombol papan ketik dan pergerakan jari.
3. Kesalahan transmisi dan penyimpanan yang berhubungan dengan pengkodean pada jalur mekanisme transmisi data.

Kesalahan ejaan dapat dikoreksi menggunakan dua strategi dasar yang berbeda, yaitu mutlak dan relatif (Pullock & Zamora 1984, dalam Wahyudin 1999). Secara mutlak, pengoreksian dilakukan dengan membuat suatu tabel variasi ejaan yang salah dengan ejaan yang benarnya. Namun demikian, secara relatif ejaan yang benar dipilih dari kamus yaitu dengan mencari kata dalam kamus yang paling mirip dengan kata yang salah ejaannya.

Damerau Levenshtein Metric adalah sebuah fungsi pada *finite* string dari sebuah alphabet ke integer. Sebuah matriks jarak yang diberikan *strings s1, s2, s3* yang memenuhi kondisi (Bard 2006):

- *Non-negativity*: $d(s_1, s_2) \geq 0$
- *Non-degeneracy*: $d(s_1, s_2) = 0$ jika dan hanya jika $s_1 = s_2$
- *Symmetry*: $d(s_1, s_2) = d(s_2, s_1)$
- *Triangle Inequality:* $d(s_1, s_2) + d(s_2, s_3) \geq d(s_1, s_3)$

Jarak $d(s_1, s_2)$ didefinisikan sebagai sebuah kombinasi operasi penjumlahan dari penambahan sebuah huruf, penghilangan sebuah huruf, penggantian sebuah huruf atau penukaran sebuah huruf dari huruf lainnya dalam satu lokasi.

Itu memungkinkan beberapa kombinasi pada empat operasi yang dapat mentransformasikan string *s1* ke *s2*, tapi panjang terpendek urutan adalah jarak antara dua *strings*.

Metode Damerau Levenshtein Metric melakukan operasi perbandingan kata-kata dengan memperhatikan empat macam kesalahan pengetikan (misalnya kata DAMERAU), yaitu:

1. Penyisipan sebuah huruf, misalnya DAHMERAU.
2. Penghapusan sebuah huruf, misalnya DAMRAU.
3. Penggantian sebuah huruf dengan huruf lain, misalnya DANERAU.
4. Penukaran sebuah huruf berurutan, misalnya DAMERUA.

Salah satu metode pembandingan dalam memeriksa ejaan dengan menggunakan *Table Look-up.* Metode ini membandingkan kata terhadap kata dalam kamus. Jika tidak ada dikamus maka kata tersebut dianggap salah (Peterson 1980, diacu dalam Wahyudin 1999). Ketidaksesuaian dari *strings* dapat dibandingkan dengan kata pada kamus yang secara langsung menggunakan penyesuaian karakter demi karakter secara iteratif, dengan menentukan jumlah minimum kesalahan dari masing-masing operasi.

Damerau Levenshtein Metric menghitung jumlah minimum kesalahan dari dua kata misalnya terdapat dua buah kata yang dinotasikan sebagai *s* dan *t*. Variabel *i* dan *j* menyatakan posisi huruf yang dibandingkan pada suatu kata (Pfiefer *et al*. 1994).

$f(0, 0) = 0$
$f(i, 0) = i$
$f(0, j) = j$
$f(i, j) = min \{$
$\quad f(i-1, j) + 1, \quad$ // *deletion*
$\quad f(i, j-1) + 1, \quad$ // *insertion*
$\quad f(i-1, j-1) + d(s_i, t_j)$ // *substitution*
$\quad f(i-2, j-2) + d(s_{i-1}, t_j) + d(s_i, t_{j-1}) + 1$ // *transposition*
$\}$

fungsi *d* merupakan fungsi untuk mengukur jarak untuk huruf.

$$f(s_i, t_j) = \begin{cases} 0, & s_i = t_j \\ 1, & s_i \neq t_j \end{cases}$$

Fungsi $f(i, j)$ menghitung minimum jumlah kesalahan-kesalahan perbandingan *i* karakter pertama dari kata pertama dengan *j* karakter pada kata kedua. Jarak antara dua *strings* adalah $f(m,n)$, dimana *m* merupakan panjang *string* pertama dan *n* merupakan panjang *string* ke dua.

Tujuan percobaan ini adalah untuk mengetahui kinerja algoritme pengoreksian ejaan Damerau Levenshtein. Akan dilihat apakah algoritme tersebut dapat memberikan usulan kata yang cukup baik untuk berbagai variasi kesalahan yang terjadi.

Percobaan dilakukan dengan mengetikkan *query* yang mengandung salah eja. Hal ini juga dilakukan seolah-olah tidak mengetahui ejaan *query* yang benar (diasumsikan ejaan *query* benar walaupun mengandung salah eja). Kemudian pada program yang telah dibuat sudah ditentukan *default* nilai jarak *edit* yaitu 2.

Asumsi-asumsi yang digunakan dalam penelitian ini antara lain:

- *Query* yang dimasukkan ke sistem merupakan kata kunci yang telah ditentukan sebelumnya.
- Jumlah dokumen relevan untuk setiap *query* telah diketahui sebelumnya.
- Teks pada dokumen benar sesuai dengan Kamus Besar Bahasa Indonesia.
- Pengoreksian hanya dilakukan pada *query*.

HASIL DAN PERCOBAAN

Dokumen korpus yang digunakan untuk pengujian sebanyak 1000 dokumen dalam bentuk *file* teks yang berformat XML. Contoh dokumen pengujian dapat dilihat pada Lampiran 1. Deskripsi dokumen pengujian ditunjukan oleh Tabel 1.

Tabel 1. Deskripsi dokumen pengujian

Uraian	Nilai (*bytes*)
Ukuran rata-rata dokumen	4.111
Ukuran dokumen keseluruhan	4.110.509
Ukuran dokumen terbesar	53.306
Ukuran dokumen terkecil	456

Percobaan dilakukan dengan *query* sebanyak 30 *query* yang telah ditentukan. Lampiran 2 merupakan contoh *query* yang digunakan sebagai percobaan. Masing-masing *query* dilakukan 3 kali percobaan. Lampiran 3 merupakan hasil percobaan untuk masing-masing percobaan jenis kesalahan. Banyaknya kata usulan yang dimunculkan tergantung banyaknya kata di kamus serta nilai jarak *edit* yang ditentukan sebelumnya.

Pada percobaan pengoreksian ejaan terdapat kata yang dikoreksi diberikan kata usulan oleh sistem tetapi tidak sesuai dengan kata yang diharapkan, seperti kata “organik” dan “kekeringan”. Karena keterbatasan kata dalam kamus, sistem hanya mengusulkan mungkin kata tersebut adalah “organnik”. Namun, kata “kekeringan”, sistem memunculkan kata usulan “kepeningan”, “kemiringan”, “keserongan”, “kekuningan”, “kekurangan”, “kekejangan”, “kebeningan”, “keserangan”, “keterangan”, “keheningan” dan “keberingas”. Padahal kata “organik” dan “kekeringan” benar sesuai ejaan bahasa Indonesia. Dengan demikian, kata “organik” dan “kekeringan” dilakukan proses penyimpan pada kamus khusus.

Untuk mengukur performansi dari temu kembali informasi, digunakan koleksi uji, koleksi uji terdiri atas tiga bagian, yaitu koleksi dokumen, *query*, dan *relevance judgement*. Koleksi dokumen adalah kumpulan dokumen yang dijadikan bahan pencarian oleh sistem. *Relevance judgement* adalah daftar dokumen-dokumen yang relevan dengan semua *query* yang telah disediakan.

Berdasarkan percobaan, akan dipaparkan nilai *precision* untuk beberapa tahap, yaitu nilai *precision* sebelum dilakukan pengoreksian, nilai *precision* setelah pengoreksian dengan memilih kata usulan yang bukan diharapkan, dan nilai *precision* setelah pengoreksian dengan kata usulan yang diharapkan. Pada paper ini nilai *precision* untuk semua percobaan akan dilihat berdasarkan rata-rata *precision* dari semua percobaan.

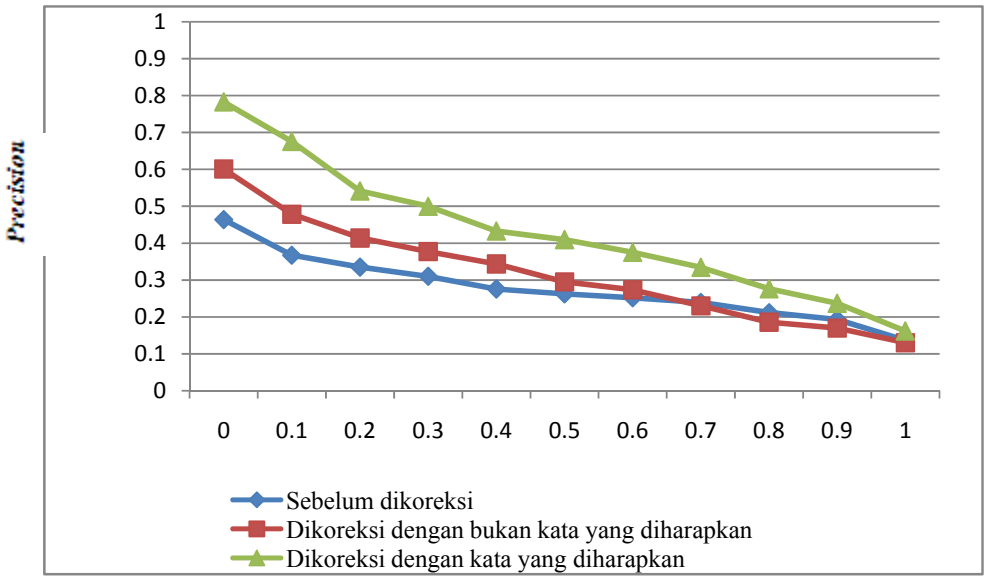
Berdasarkan Tabel 2 dan Tabel 3, nilai *precision* untuk tahap sebelum dilakukan pengoreksian, tahap setelah pengoreksian dengan memilih kata usulan yang bukan diharapkan, dan tahap setelah pengoreksian dengan kata usulan yang diharapkan diperoleh dengan rata-rata nilai *precison* dari semua percobaan. Nilai rata-rata *precision* setelah tahap setelah pengoreksian dengan kata usulan yang diharapkan jauh lebih tinggi dibanding dengan tahap sebelum dilakukan pengoreksian maupun tahap setelah pengoreksian dengan kata usulan yang diharapkan. Hal ini disebabkan, *query* setelah tahap setelah pengoreksian dengan kata usulan yang diharapkan menjadi *query* yang lebih optimal. Sementara itu, grafik perbandingan *precision* tahap sebelum dilakukan pengoreksian, tahap setelah pengoreksian dengan memilih kata usulan yang bukan diharapkan, dan tahap setelah pengoreksian dengan kata usulan yang diharapkan dengan melihat minimum *precision* dapat dilihat pada Gambar 1.

Tabel 2. Perbandingan rata-rata nilai *precision* tahap sebelum dikoreksi dan tahap setelah pengoreksian dengan kata yang bukan diharapkan

recall	**Sebelum dikoreksi**	**Dikoreksi dengan bukan kata yang diharapkan**	**Perubahan (%)**
0	0,48	0,60	25
0,1	0,38	0,48	26
0,2	0,35	0,41	17
0,3	0,32	0,38	19
0,4	0,28	0,34	21
0,5	0,27	0,29	7
0,6	0,26	0,27	4
0,7	0,25	0,23	-8
0,8	0,22	0,19	-14
0,9	0,20	0,17	-15
1	0,14	0,13	-7
Rataan	**0,28**	**0,32**	**6,82**

Tabel 3. Perbandingan rata-rata nilai *precision* tahap sebelum dikoreksi dan setelah pengoreksian dengan kata yang diharapkan

recall	**Sebelum dikoreksi**	**Dikoreksi dengan kata yang diharapkan**	**Perubahan (%)**
0	0,48	0,78	63
0,1	0,38	0,68	79
0,2	0,35	0,54	54
0,3	0,32	0,50	56
0,4	0,28	0,43	54
0,5	0,27	0,41	52
0,6	0,26	0,37	42
0,7	0,25	0,33	32
0,8	0,22	0,28	27
0,9	0,20	0,24	20
1	0,14	0,16	14
Rataan	**0,28**	**0,43**	**44,82**



Gambar 1. Kurva rata-rata *recall-precision* tahap sebelum pengoreksian, tahap setelah pengoreksian dengan bukan kata diharapkan, dan tahap setelah pengoreksian dengan kata yang diharapkan.

Kurva perbandingan *recall-precision* pada Gambar 1 menunjukkan bahwa kurva *recall-precision* setelah tahap setelah pengoreksian dengan kata yang diharapkan selalu berada di atas dua kurva tahap sebelum pengoreksian dan tahap setelah pengoreksian dengan bukan kata diharapkan. Hal ini terjadi peningkatan nilai rata-rata *precision* yang baik setiap titik *recall*. Ini menunjukkan *query* setelah dilakukan pengoreksian dengan kata yang diharapkan jauh lebih optimal. *Query* yang lebih optimal menyebabkan peningkatan dokumen-dokumen relevan yang ditemukembalikan.

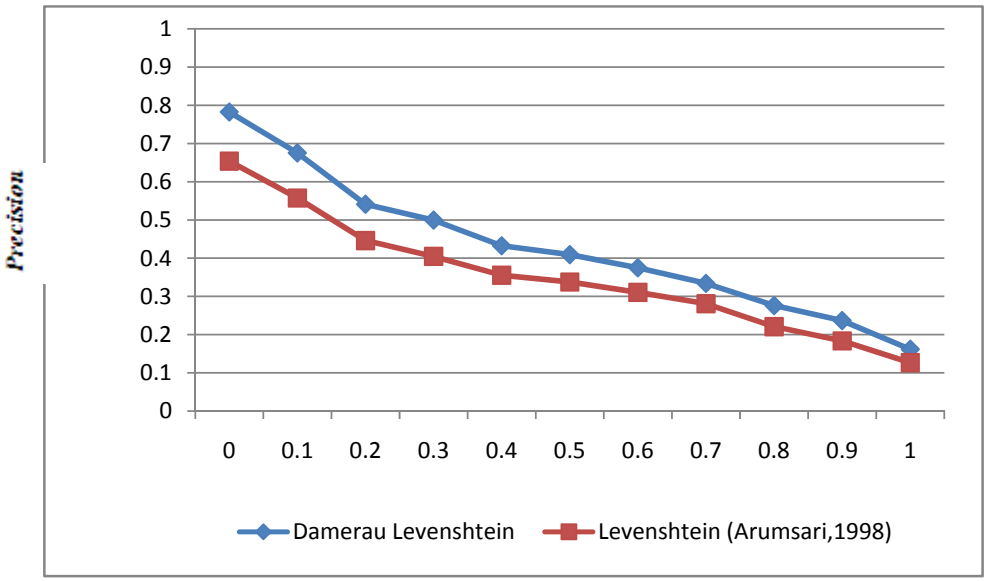
Pengoreksian ejaan *query* dengan menggunakan algoritme Damerau Levenshtein dalam percobaan, menunjukkan hasil yang sangat baik. Ini terlihat secara rata-rata perbandingan kurva sebelum *query* dikoreksi dan setelah dikoreksi dengan kata yang diharapkan terlihat nilai *precision* temu kembali meningkat untuk setiap titik *recall*. Ini juga menunjukkan dokumen-dokumen relevan yang ditemukembalikan berada pada halaman pertama dari sistem temu kembali.

Perbandingan Kinerja Temu Kembali dengan Levenshtein (Arumsari 1998)

Dari hasil percobaan evaluasi temu kembali yang dilakukan dengan menggunakan algoritme Damerau Levenshtein. Tabel 3 menunjukkan nilai *precision* perbandingan pengoreksian dengan menggunakan algoritme Damerau Levenshtein dan jarak *edit* (Arumsari 1998).

Tabel 3. Perbandingan nilai *precision* dengan algoritme Damerau Levenshtein dengan metode Levenshtein (Arumsari 1998)

recall	**Levenshtein**	**Damerau Levenshtein**	**Perubahan (%)**
0	0,65	0,78	20
0,1	0,56	0,68	21
0,2	0,45	0,54	20
0,3	0,4	0,5	25
0,4	0,36	0,43	19
0,5	0,34	0,41	21
0,6	0,31	0,37	19
0,7	0,28	0,33	18
0,8	0,22	0,28	27
0,9	0,18	0,24	33
1	0,13	0,16	23
Rataan	**0,35**	**0,43**	**22,36**



Gambar 2. Kurva *recall-precision* perbandingan algoritme Damerau Levenshtein dengan Levenshtein.

Berdasarkan Gambar 2, kurva *recall-precision* dengan algoritme Damerau Levenshtein berada di atas kurva dengan metode Levenshtein (Arumsari 1998) untuk setiap titik *recall*-nya. Ini menunjukkan pengoreksian ejaan dengan algoritme Damerau Levenshtein lebih optimal dan meningkatkan dokumen yang ditemukembalikan pada sistem. Ini terlihat ketika pengoreksian ejaan seperti pada kata "perdagagnang", "labroatoruim", "haisl" dengan menggunakan Levenshtein (Arumsari 1998), sistem tidak menemukan kata usulan atau tidak memberikan kata usulan yang diharapkan.

KESIMPULAN DAN SARAN

Melalui penelitian ini dapat ditarik kesimpulan, yaitu :

1. Implementasi algoritme Damerau Levenshtein untuk koreksi ejaan pada *search engine*, dapat meningkatkan kinerja temu kembali dan *query* menjadi lebih optimal. Ini terlihat peningkatan secara rata-rata *precision* sebesar 44,82 % setelah dilakukan pengoreksian.
2. Pengoreksian dengan algoritme Damerau Levenshtein lebih baik dibanding dengan metode jarak *edit* yang dilakukan oleh Arumsari (1998). Ini terlihat peningkatan *precision* pada kinerja temu kembali sebesar 22%.

DAFTAR PUSTAKA

Arumsari, KN. 1998. Penggunaan Metode Kesamaan String pada Pemeriksaan Ejaan Bahasa Indonesia *[skripsi]*. Bogor: Departemen Ilmu Komputer, Fakultas MIPA, Institut Pertanian Bogor.

Bard GV. 2006. *Spelling-Error Tolerant, Order-Independent Pass-Phrases via the Damerau-Levenshtein String-Edit Distance Metric*. University of Maryland.

Primasari D. 1997. Metode Pencarian dan Temu Kembali Nama Berdasarkan Kesamaan Fonetik. *[skripsi]*. Bogor: Departemen Ilmu Komputer, Fakultas MIPA, Institut Pertanian Bogor.

Manning CD, Prabhakar R, Hinrich S. 2009. An *Introduction to Information Retrieval*. Cambridge University Press.

Pfeifer U, Poersch T, Furh N. 1994. *Searching Proper Names in Databases*. University of Dortmund. *Wahyudin, Aep. 1999. Algoritme Trigram untuk Mengoreksi Ejaan [skripsi]*. Bogor: Departemen Ilmu Komputer, Fakultas MIPA, Institut Pertanian Bogor.

Wahyudin, Aep. 1999. Algoritme Trigram untuk Mengoreksi Ejaan [skripsi]. Bogor: Departemen Ilmu Komputer, Fakultas MIPA, Institut Pertanian Bogor