Pencocokan *String* Berdasarkan Kemiripan Ucapan (*Phonetic String Matching*) dalam Bahasa Inggris

Mokhamad Syaroni [1], Rinaldi Munir [2]

Laboratorium Ilmu dan Rekayasa Komputasi
Departemen Teknik Informatika, Fakultas Teknologi Industri, Institut Teknologi Bandung (ITB)
Kampus ITB Jl Ganesha No. 10 Bandung
if11064@students.if.itb.ac.id [1], rinaldi@informatika.org [2]

Abstrak

Perkembangan teknologi informasi dan komunikasi yang semakin pesat, mengakibatkan pencarian informasi semakin mudah. Pencarian informasi yang tepat dan sesuai kebutuhan menjadi sangat penting. Oleh karena itu, teknik untuk memperoleh dokumen dengan isi yang sesuai dengan kebutuhan informasi sangat diperlukan. Untuk mengetahui isi dokumen sesuai dengan kebutuhan informasi, diperlukan metode pencarian *string* (*string searching*) isi dokumen yang bagus. Proses pencocokan *string* (*string matching*) yang merupakan bagian dalam proses pencarian *string* memegang peranan penting untuk mendapatkan dokumen yang sesuai dengan kebutuhan informasi. Pencocokan *string* (*string matching*) secara garis besar dapat dibedakan menjadi dua yaitu pencocokan *string* secara eksak/sama persis (*exact string matching*) dan pencocokan *string* berdasarkan kemiripan (*inexact string matching/fuzzy string matching*). Pencocokan string berdasarkan kemiripan masih dapat dibedakan menjadi dua yaitu berdasarkan kemiripan penulisan (*approximate string matching*) dan berdasarkan kemiripan ucapan (*phonetic string matching*). Contoh *phonetic string matching* adalah kata *step* akan menunjukkan kecocokan dengan kata *step*, *sttep, stepp*, *sstep, stepe, steb.* Sedangkan bila kita menggunakan *exact string matching* kata *step* hanya akan menunjukkan kecocokan dengan kata *step* saja. Pada makalah ini akan dibahas dan dianalisis kemampuan tiga algoritma *phonetic string matching* yaitu algoritma *soundex*, *metaphone,* dan *caverphone* dari segi fonetik bahasa Inggris.

***Kata kunci**: pencocokan string, string matching, inexact string matching, phonetic string matching, soundex, metaphone, caverphone*

1. Pendahuluan

Pencarian informasi yang tepat dan sesuai kebutuhan menjadi sangat penting dengan semakin mudahnya memperoleh informasi dari seluruh dunia sebagai akibat perkembangan teknologi informasi dan komunikasi yang semakin pesat. Oleh karena itu, teknik untuk memperoleh dokumen dengan isi yang sesuai dengan kebutuhan informasi sangat diperlukan. Untuk mengetahui isi dokumen yang benar-benar sesuai dengan kebutuhan informasi, diperlukan metode pencarian *string* (*string searching*) isi dokumen yang bagus. Proses pencocokan *string* (*string matching*) yang merupakan bagian utama dalam proses pencarian *string* memegang peranan penting untuk mendapatkan dokumen yang sesuai dengan kebutuhan informasi tersebut.

2. Pencocokan *String* (*String Matching*)

2.1 Pengertian Pencocokan *String*

Pengertian *string* menurut *Dictionary of Algorithms and Data Structures, National Institute of Standards and Technology (NIST)* adalah susunan dari karakter-karakter (angka, alfabet atau karakter yang lain) dan biasanya direpresentasikan sebagai struktur data *array*. *String* dapat berupa kata, frase, atau kalimat. [3]

Pencocokan *string* merupakan bagian penting dari sebuah proses pencarian *string* (*string searching*) dalam sebuah dokumen. Hasil dari pencarian sebuah *string* dalam dokumen tergantung dari teknik atau cara pencocokan *string* yang digunakan.

Pencocokan *string (string matching)* menurut *Dictionary of Algorithms and Data Structures, National Institute of Standards and Technology (NIST),* diartikan sebagai sebuah permasalahan untuk menemukan pola susunan karakter *string* di dalam *string* lain atau bagian dari isi teks [3].

2.2 Klasifikasi Pencocokan *String*

Pencocokan *string* (*string matching*) secara garis besar dapat dibedakan menjadi dua yaitu : [1]

1. *Exact string matching*, merupakan pencocokan *string* secara tepat dengan susunan karakter dalam *string* yang dicocokkan memiliki jumlah maupun urutan karakter dalam *string* yang sama. Contoh : kata *step* akan menunjukkan kecocokan hanya dengan kata *step*.
2. *Inexact string matching* atau *Fuzzy string matching*, merupakan pencocokan *string* secara samar, maksudnya pencocokan *string* dimana *string* yang dicocokkan memiliki kemiripan dimana keduanya memiliki susunan karakter yang berbeda (mungkin jumlah atau urutannya) tetapi *string-string* tersebut memiliki kemiripan baik kemiripan tekstual/penulisan (*approximate string matching*) atau kemiripan ucapan (*phonetic string matching*). *Inexact string matching* masih dapat dibagi lagi menjadi dua yaitu :
   a. Pencocokan *string* berdasarkan kemiripan penulisan (*approximate string matching*) merupakan pencocokan *string* dengan dasar kemiripan dari segi penulisannya (jumlah karakter, susunan karakter dalam dokumen). Tingkat kemiripan ditentukan dengan jauh tidaknya beda penulisan dua buah *string* yang dibandingkan tersebut dan nilai tingkat kemiripan ini ditentukan oleh pemrogram (*programmer*). Contoh : *c mpuler* dengan *compiler*, memiliki jumlah karakter yang sama tetapi ada dua karakter yang berbeda. Jika perbedaan dua karakter ini dapat ditoleransi sebagai sebuah kesalahan penulisan maka dua *string* tersebut dikatakan cocok.
   b. Pencocokan *string* berdasarkan kemiripan ucapan (*phonetic string matching*) merupakan pencocokan *string* dengan dasar kemiripan dari segi pengucapannya meskipun ada perbedaan penulisan dua *string* yang dibandingkan tersebut. Contoh *step* dengan *steb* dari tulisan berbeda tetapi dalam pengucapannya mirip sehingga dua *string* tersebut dianggap cocok. Contoh yang lain adalah *step*, dengan *steppe*, *sttep*, *stepp*, *stepe*.

Exact string matching bermanfaat jika pengguna ingin mencari *string* dalam dokumen yang sama persis dengan *string* masukan. Tetapi jika pengguna menginginkan pencarian *string* yang mendekati dengan *string* masukan atau terjadi kesalahan penulisan *string* masukan maupun dokumen objek pencarian, maka *inexact string matching* yang bermanfaat. Beberapa algoritma *exact string matching* antara lain : algoritma *Knuth-Morris-Pratt*, *Bayer-Moore*, dll. Beberapa algoritma *phonetic string matching* antara lain : *soundex*, *metaphone*, *caverphone*, *phonex*, *NYSIIS*, *Jaro-Winkler*, dll.

Dari contoh yang diberikan, sebenarnya *phonetic string matching* juga dapat dimanfaatkan untuk *approximate string matching* dengan batasan dua *string* yang dicocokkan masih memiliki kemiripan ucapan. *Phonetic string matching* sering juga dimanfaatkan untuk *approximate string matching* karena *phonetic string matching* lebih mudah diimplementasikan.

Phonetic string matching banyak digunakan dalam bahasa Inggris karena dalam bahasa Inggris terdapat perbedaan antara penulisan dan pengucapan.

3. Fonetik Bahasa Inggris

3.1. Pengertian Fonetik

Fonetik (*phonetics*) adalah ilmu yang menyelidiki bunyi bahasa tanpa melihat fungsi bunyi itu sebagai pembeda makna dalam suatu bahasa (*language*) [7]. Kata sifat fonetik adalah fonetis (*phonetic*).

Bagian fonetik bahasa Inggris yang berkaitan erat dengan *phonetic string matching* adalah klasifikasi konsonan, karena klasifikasi konsonan yang memegang peranan penting dalam *phonetic string matching*.

3.2 Klasifikasi Konsonan

Konsonan didasarkan pada alat bicara yang menghasilkannya dapat dibedakan menjadi tujuh kelompok yaitu : [6]

1. *Labial* atau bunyi bibir, yang dapat dibedakan lagi menjadi dua golongan yaitu :
   1.1. *Bilabial*, bunyi diartikulasi oleh dua bibir, contoh bunyi p, b, m, w.
   1.2. *Labio-dental*, bunyi diartikulasi oleh bibir bawah dan gigi atas, f,v.
2. *Dental*, bunyi diartikulasi oleh ujung lidah dengan gigi atas, contoh bunyi th (dalam kata *thin*)
3. *Alveolar*, bunyi diartikulasi oleh ujung lidah dengan punggung gigi (*teeth-ridge*), contoh bunyi d, t, n, l, r, s, z.
4. *Palato-alveolar*, bunyi yang memiliki artikulasi *alveolar* dikuti dengan naiknya lidah sampai pada langit-langit mulut secara simultan, contoh bunyi c, j, sh (dalam kata *show*).
5. *Palatal*, bunyi diartikulasi oleh bagian depan lidah dengan langit-langit keras (*hard palate*), contoh bunyi y.
6. *Velar*, bunyi diartikulasi oleh bagian belakang lidah dengan langit-langit lunak (*soft palate*), contoh bunyi k, g.
7. *Glottal*, bunyi diartikulasi oleh *glottis*, contoh bunyi h.

Berdasarkan cara artikulasi (dihambat), konsonan dibedakan menjadi delapan golongan yaitu : [6]

1. Konsonan hambat letup (*plosive*), merupakan konsonan yang terjadi dengan hambatan penuh arus udara kemudian hambatan itu dilepaskan secara tiba-tiba. Contoh : b, d, g, k, p, t.
2. Konsonan nasal atau sengau (*nasal*), merupakan konsonan yang dibentuk dengan menghambat rapat (menutup) jalan udara dari paru-paru melalui rongga mulut, bersama dengan itu langit-langit lunak beserta anak tekaknya diturunkan sehingga udara keluar melalui rongga hidung. Contoh : m, n.
3. Konsonan paduan (*affricate*), merupakan konsonan hambat jenis khusus dimana proses terjadinya dengan menghambat penuh arus udara dari paru-paru, kemudian hambatan itu dilepaskan secara pelan-pelan. Tempat artikulasinya pada *palato-alveolar*. Contoh : c, j.
4. Konsonan sampingan (*lateral*), merupakan konsonan yang dibentuk dengan menutup arus udara di tengah rongga mulut sehingga udara keluar melalui kedua samping atau sebuah samping saja. Tempat artikulasinya pada *alveolar*. Contoh : l.
5. Konsonan geseran (*fricative*), merupakan konsonan yang dibentuk dengan menyempitkan jalannya arus udara yang dihembuskan dari paru-paru, sehingga jalannya udara terhalang dan keluar dengan bergeser. Contoh : f, v, r, s, z, th (dalam kata *thin*), sh (dalam kata *show*), h.
6. Konsonan getar (*rolled*), merupakan konsonan yang dibentuk dengan menghambat jalannya arus udara yang dihembuskan dari paru-paru secara berulang-ulang dan cepat dan terjadi banyak sentuhan (*tap*) yang terjadi antara ujung lidah dengan langit-langit atau gusi belakang. Contoh *rolled* r (sangat jarang ditemukan).
7. Konsonan sentuhan kuat (*flapped*), merupakan konsonan dengan proses yang hampir sama dengan konsonan *rolled* tetapi hanya terjadi satu sentuhan (*tap*) antara ujung lidah dengan langit-langit atau gusi belakang. Contoh *flapped* r (sangat jarang ditemukan).
8. Semi vokal (*semi-vowels*), bunyi yang secara praktis termasuk konsonan tetapi karena pada waktu diartikulasikan belum membentuk konsonan murni, maka bunyi-bunyi itu disebut semi-vokal. Contoh : w, y.

Tabel 1 menunjukkan pembagian konsonan bahasa Inggris berdasarkan alat bicara yang menghasilkan dan cara artikulasinya :

Tabel 1. Konsonan Bahasa Inggris [6]

Klasifikasi Konsonan	*Labial*		*Dental*	*Alveolar*	*Palato-Alveolar*	*Palatal*	*Velar*	*Glottal*
	Bilabial	*Labiodental*						
Plosive	b, p			t, d			k, g	
Affricate					c,j			
Nasal	m			n				
Lateral				l				
Rolled								
Flapped								
Fricative		f, v	th	s, z, r	sh			h
Semi-vowel	w					y		

4. Algoritma *Phonetic String Matching*

Algoritma *phonetic string matching* yang akan dibahas dan dianalisis dalam makalah ini meliputi algoritma *soundex*, *metaphone*, dan *caverphone*. Alasan dipilihnya ketiga algoritma ini karena :

1. Algoritma *soundex* merupakan algoritma yang paling banyak digunakan.
2. Algoritma *metaphone* merupakan algoritma yang melakukan penanganan secara khusus terhadap setiap fonem (satuan bunyi bahasa) dalam kata.
3. Algoritma *caverphone* merupakan algoritma yang masih baru dan berusaha menyempurnakan algoritma-algoritma terdahulu.
4. Ketiga algoritma memiliki langkah umum yang sama dalam mencocokkan kata.

Langkah umum ketiga algoritma dalam mencocokkan kata meliputi :

1. Menerima *input* berupa dua *string* yang akan dicocokkan
2. Mengubah dua *string* masukan menjadi dua buah kode fonetis (*phonetic code*). Kode fonetis (*phonetic code*) adalah kode yang dihasilkan dari sebuah *string* berdasarkan cara pengucapannya [3].
3. Membandingkan dua buah kode fonetis yang dihasilkan, jika kode fonetis sama maka dua *string* dianggap cocok (mirip cara pengucapan), jika kode fonetis berbeda maka dua *string* dianggap tidak cocok.

4.1 Algoritma *Soundex*

Algoritma *soundex* pertama kali dipatenkan oleh Margaret O'Dell and Robert C. Russell pada tahun 1918 [5], tetapi algoritma ini kemudian disempurnakan lagi [4]. Algoritma *soundex* menghasilkan kode fonetik dengan panjang empat karakter untuk semua panjang *string* masukan.

Langkah-langkah algoritma soundex dalam menghasilkan kode fonetis dapat dilihat pada http://www.creativyst.com/Doc/Articles/SoundEx1/SoundEx1.htm [4]. Aturan pemberian kode fonetis dalam algoritma *soundex* dapat dilihat pada tabel 2 dan tabel 3.

Tabel 2. Aturan Baru *Soundex* [4]

Awal	Akhir	Keterangan
DG	G	
GH	H	
GN	N	
KN	N	
PH	F	
MP	M	Ketika diikuti S, Z, T
PS	S	Ketika di awal kata
PF	F	Ketika di awal kata
MB	M	
TCH	CH	
A	E	Ketika kata diawali A+[A,E,I,O]
I	E	Ketika kata diawali A+[A,E,I,O]

Tabel 3. Aturan Pemberian Kode pada Algoritma *Soundex*

Huruf-Huruf	Kode
'A', 'E', 'H', 'I', 'O', 'U', 'W', 'Y'	'0'
'B', 'F', 'P', 'V'	'1'
'C', 'G', 'J', 'K', 'Q', 'S', 'X', 'Z'	'2'
'D', 'T'	'3'
'L'	'4'
'M', 'N'	'5'
'R'	'6'

Prinsip dasar algoritma *soundex* di atas berdasarkan klasifikasi konsonan menurut alat bicara yang menghasilkannya dan cara artikulasinya. Penjelasan cara pemberian kode angka terhadap huruf-huruf dalam kata dilihat dari segi fonetik bahasa Inggris adalah :

1. Kode '0' untuk huruf-huruf 'a', 'e', 'h', 'i', 'o', 'u', 'w', 'y' disebabkan huruf-huruf tersebut diperlakukan sebagai bunyi vokal..
2. Huruf-huruf 'b', 'f', 'p', 'v' dikelompokkan menjadi satu dengan kode '1' karena terdapat dalam satu kelompok konsonan yaitu konsonan *labial*.
3. Huruf-huruf 'c', 'g', 'j', 'k', 'q', 's', 'x', 'z' dikelompokkan menjadi satu dengan kode '2' karena pengucapan huruf-huruf tersebut mirip.
4. Huruf- huruf 'd', 't' dikelompokkan menjadi satu dengan kode '3' karena terdapat dalam satu kelompok konsonan yaitu *plosive alveolar*.
5. Huruf 'l' dikelompokkan tersendiri menjadi satu dengan kode '4' karena merupakan konsonan *lateral*.
6. Huruf-huruf 'm', 'n' dikelompokkan menjadi satu dengan kode '5' karena huruf 'm' dan 'n' keduanya merupakan konsonan *nasal*.
7. Huruf 'r' pada kedua algoritma *soundex* di atas dikelompokkan tersendiri menjadi satu kelompok dan diberi kode '6' karena huruf 'r' dalam pengucapan bahasa Inggris berbeda dengan konsonan lain dan sangat beragam karena ada *flapped* r (termasuk konsonan *flapped*), *rolled* r (termasuk konsonan *rolled*), dan masuk konsonan *fricative*.

4.2 Algoritma Metaphone

Algoritma *metaphone* dipublikasikan oleh Lawrence Philips dalam sebuah artikel berjudul "*Hanging on the Metaphone*" dalam jurnal "*Computer Language*" vol. 7 n. 12, Desember 1990, pp. 39-43. Algoritma *metaphone* menghasilkan kode fonetis yang panjang karakternya berbeda sesuai dengan panjang *string* masukan [1].

Langkah-langkah dalam algoritma dapat dilihat pada buku "*Practical Algorithms for Programmers*" [1]. Pada algoritma *metaphone* vokal diabaikan kecuali pada awal kata. Dalam algoritma ini, mereduksi alfabet menjadi enam belas suara konsonan yaitu : B, X, S, K, J, T, F, H, L, M, N, P, R, 0, W, dan Y. Suara 'sh' direpresentasikan dengan 'X' dan nol ('0') merepresentasikan suara 'th'. Aturan pemberian kode fonetis dapat dilihat pada tabel 4.

Tabel 4 Aturan *Metaphone* [1]

Awal	Akhir	Keterangan
B	B	Kecuali di akhir kata setelah 'M' Contoh : *dumb*
C	X	(sh), jika dalam '-CIA-','-CH-'
	S	Jika dalam '-CI-', '-CE-', '-CY-'
	dihapus (*silent*)	Jika dalam '-SCI-', '-SCE-', '-SCY-'
	K	
D	J	Jika dalam '-DGE-', '-DGY-', 'DGI'
	T	
F	F	
G	F	Jika dalam '-GH' kecuali dalam 'B--GH', 'D--GH', '-H--GH','-H---GH'
	dihapus (silent)	Jika dalam '-GNED', '-GN', '-DGE-', '-DGI- '-DGY-'
	J	Jika dalam '-GE', '-GI','-GY' dan tidak dalam 'GG'
	K	
H	dihapus (silent)	Jika sesudah vokal dan tidak diikuti vokal
	H	Jika sebelum sebuah vokal dan tidak sesudah 'C', 'G', 'P', 'S', 'T'
J	J	
K	dihapus (silent)	Jika sesudah 'C'
	K	
L	L	
M	M	
N	N	
P	F	Jika sebelum 'H'
	P	
Q	K	
R	R	
S	X	(sh), jika dalam '-SIA-','-SIO-' atau sebelum 'H'
	S	
T	X	(sh), jika dalam '-TIA-','-TIO-'
	0	(th), jika sebelum 'H'
	dihapus (silent)	Jika di dalam '-TCH-'
	T	
V	F	
W	W	Jika sesudah vokal
	dihapus (silent)	
X	KS	
Y	Y	Jika sesudah vokal
	dihapus (silent)	
Z	S	

Pada algoritma *metaphone*, pemberian kode fonetis memperhatikan juga interaksi antara konsonan dan vokal dalam kata serta kelompok konsonan bukan hanya sebuah konsonan seperti pada algoritma *soundex*. Walaupun pada algoritma *soundex* sudah diberikan aturan baru mengenai perlakuan terhadap kelompok konsonan sebelum langkah pemberian kode fonetis, tetapi masih sangat terbatas.

4.3 Algoritma Caverphone

Algoritma *caverphone* dikembangkan oleh David Hood dalam proyek Caversham, Universitas Otago, New Zealand. Pertama kali, algoritma *caverphone* diperkenalkan pada tahun 2002 dan ditujukan untuk pencocokan nama orang saja, tetapi kemudian dikembangkan lagi sehingga muncul algoritma *caverphone* versi 2.0 pada tahun 2004 yang lebih umum untuk pencocokan kata dalam bahasa Inggris.

Pada algoritma *caverphone* versi 2.0 menghasilkan kode fonetis dengan panjang sepuluh karakter [2].

Seperti algoritma *soundex* dan *metaphone*, algoritma *caverphone* juga mengelompokkan huruf-huruf yang cenderung menimbulkan kesalahan pengucapan (satu kelompok dalam klasifikasi konsonan) dalam satu kelompok, contoh p/b, t/d.

Langkah-langkah algoritma *caverphone* versi 2.0 dalam pemberian kode fonetis dapat dilihat pada http://caversham.otago.ac.nz [2]. Perbedaan yang tampak antara algoritma *caverphone* dengan algoritma *soundex* dan algoritma *metaphone*, jika kita memperhatikan penjelasan algoritma-algoritma tersebut dari segi fonetis adalah :

1. Algoritma *caverphone* mempunyai langkah-langkah yang panjang dan rinci serta tidak menangani khusus untuk huruf-hurf tertentu, tetapi sudah digabung untuk beberapa huruf.
2. Algoritma *caverphone* dirancang agar peka terhadap letak huruf-huruf dalam kata.

Algoritma *caverphone* ini masih dikembangkan untuk mendapatkan algoritma *phonetic string matching* yang lebih baik kemampuannya dalam pencocokan dari segi fonetik bahasa Inggris dan memiliki langkah yang lebih sedikit.

5. Analisis Kemampuan Pencocokan Tiga Algoritma yang Dikaji

Analisis kemampuan pencocokan kata untuk ketiga algoritma yang dikaji dititikberatkan pada kemampuan *phonetic string matching*, tetapi selain itu juga dilakukan analisis kemampuan ketiga algoritma untuk *approximate string matching*. Aspek yang akan dianalisis meliputi :

1. Pengaruh pengubahan konsonan yang masih dalam satu kelompok klasifikasi konsonan (mirip pengucapannya) terhadap kemampuan pencocokan
2. Pengaruh pengubahan vokal terhadap kemampuan pencocokan
3. Pengaruh panjang kata yang dicocokkan
4. Pengaruh pengubahan panjang kode fonetis terhadap kemampuan pencocokan
5. Pengaruh keberadaan karakter khusus dalam kata terhadap kemampuan pencocokan
6. Kemampuan *retrieval* dalam pencarian kata
7. Kemampuan *approximate string matching*.

Untuk melakukan analisis kemampuan pencocokan kata ketiga algoritma tersebut, dibuat sebuah perangkat lunak **PhoneMatching**.

Pengujian pengubahan konsonan yang masih dalam satu kelompok klasifikasi konsonan terhadap kemampuan pencocokan, dilakukan untuk beberapa posisi konsonan dalam kata. Hasil pengujian dapat dilihat pada tabel 5.

Tabel 5. Pengujian terhadap Pengelompokan Konsonan

Kelompok Konsonan	Algoritma								
	Soundex			*Metaphone*			*Caverphone*		
	awal	tengah	akhir	awal	tengah	akhir	awal	tengah	akhir
Plosive bilabial	x	y	y	x	x	x	y	y	y
Plosive alveolar	x	y	y	y	y	y	y	y	y
Plosive velar	x	y	y	x	x	y	y	y	y
Affricate palato-alveolar	x	y	y	x	x	x	x	x	x
Fricative labiodental	x	y	y	y	y	y	y	y	y
Fricative alveolar	x	y	y	y	y	y	y	y	y

Keterangan : x = tidak cocok, y = cocok

Hasil pengujian diatas dapat disimpulkan beberapa hal antara lain :

1. Algoritma *soundex* sangat bergantung pada huruf awal kata dalam pencocokan kata, walaupun konsonan mirip pengucapannya (satu kelompok) dalam dua buah kata yang dicocokkan tetap tidak mampu mencocokkan kalau konsonan terletak pada awal kata.
2. Untuk ketiga algoritma yang dikaji, kemampuan *phonetic string matching* dari aspek pengelompokan konsonan yang paling baik ditunjukkan oleh algoritma *caverphone*.

Pengujian pengaruh pengubahan vokal terhadap kemampuan pencocokan dilakukan untuk beberapa posisi vokal dalam kata. Hasil pengujian dapat dilihat pada tabel 6.

Tabel 6. Pengujian terhadap Pengaruh Pengubahan Vokal

Pengubahan vokal		Algoritma								
mula	akhir	*soundex*			*metaphone*			*caverphone*		
		awal	tengah	akhir	awal	tengah	akhir	awal	tengah	akhir
a	e	x	y	y	x	y	y	y	y	x
a	i	x	y	y	x	y	y	y	y	y
a	o	x	y	y	x	y	y	y	y	y
a	u	x	y	y	x	y	y	y	y	y
e	a	x	y	y	x	y	y	y	y	x
e	i	x	y	y	x	y	y	y	y	x
e	o	x	y	y	x	y	y	y	y	x
e	u	x	y	y	x	y	y	y	y	x
i	a	x	y	y	x	y	y	y	y	y
i	e	x	y	y	x	y	y	y	y	x
i	o	x	y	y	x	y	y	y	y	y
i	u	x	y	y	x	y	y	y	y	y
o	a	x	y	y	x	y	y	y	y	y
o	e	x	y	y	x	y	y	y	y	x
o	i	x	y	y	x	y	y	y	y	y
o	u	x	y	y	x	y	y	y	y	y
u	a	x	y	y	x	y	y	y	y	y
u	e	x	y	y	x	y	y	y	y	x
u	i	x	y	y	x	y	y	y	y	y
u	o	x	y	y	x	y	y	y	y	y

Keterangan : x = tidak cocok, y = cocok

Hasil pengujian diatas dapat disimpulkan beberapa hal antara lain :

1. Algoritma *soundex* dan *metaphone* memperhitungkan vokal pada awal kata, dan hal ini sudah benar karena vokal pada awal kata sangat berpengaruh. Sedangkan algoritma *caverphone* kurang memperhatikan vokal pada awal kata justru pada akhir kata, padahal vokal pada akhir kata cenderung kurang berpengaruh dalam pengucapan pada bahasa Inggris.
2. Untuk ketiga algoritma yang dikaji, kemampuan *phonetic string matching* dari aspek pengubahan

vokal yang baik, ditunjukkan oleh algoritma *soundex* dan *metaphone*.

Hasil pengujian pengaruh panjang kata yang dicocokkan menunjukkan bahwa :

1. *Phonetic string matching* sebaiknya digunakan untuk pencocokan kata-kata dengan panjang lebih dari lima karakter atau terdiri lebih dari satu suku kata. Untuk kata-kata dengan panjang lima karakter kurang atau terdiri satu suku kata saja cenderung menunjukkan kecocokan dua buah kata yang memang memiliki kemiripan ucapan tetapi memiliki perbedaan makna, dan hal ini sudah melanggar konsep fonetik
2. Untuk ketiga algoritma yang dikaji, kemampuan pencocokan kata dengan panjang lebih dari lima karakter atau terdiri lebih dari satu suku kata, ditunjukkan oleh algoritma *caverphone*.

Hasil pengujian pengaruh panjang kode fonetis menunjukkan bahwa panjang kode fonetis sangat berpengaruh terhadap kemampuan pencocokan. Panjang kode fonetis empat karakter mampu menunjukkan bahwa dua kata yang dicocokkan memiliki bentuk dasar yang sama dan hal itu dianggap sebagai kemiripan, walaupun ada perbedaan pengucapannya jika keseluruhan kata dicocokkan. Contoh kata *recognize, recognized, recognizeable, recognizably* akan menunjukkan kecocokan dengan panjang kode fonetis empat karakter. Panjang kode fonetis yang pendek mengabaikan suku kata yang terletak di akhir kata. Sedangkan panjang kode fonetis yang panjang memperhatikan suku kata yang terletak di akhir kata tetapi memiliki kelemahan tidak mampu mengelompokkan kata-kata dengan bentuk dasar sama dalam satu kelompok.

Hasil pengujian pengaruh karakter khusus dalam kata menunjukkan bahwa karakter khusus dalam kata dapat ditangani dengan baik oleh ketiga algoritma yang dikaji dan tidak mempengaruhi kemampuan pencocokan. Contoh kasus karakter apostrof (‘) yang sangat sering terdapat dalam kata bahasa Inggris misalnya *receiv’d, recogniz’d,* dll.

Hasil pengujian kemampuan *retrieval* dalam pencarian kata menunjukkan bahwa kemampuan *retrieval* dalam pencarian kata yang paling baik ditunjukkan oleh algoritma *soundex*. Hal ini disebabkan algoritma *soundex* menggunakan panjang kode fonetis lebih pendek dibanding algoritma *metaphone* dan *caverphone* yaitu empat karakter.

Hasil pengujian kemampuan *approximate string matching* untuk ketiga algoritma yang dikaji menunjukkan bahwa ketiga algoritma *phonetic string matching* yang dikaji dapat digunakan untuk *approximate string matching*. Tetapi penggunaan tersebut dengan catatan perbedaan penulisan yang ada tidak menyebabkan perubahan pengucapan terutama dalam perbedaan penulisan konsonan. Untuk perbedaan penulisan vokal, vokal yang tidak terletak di awal kurang berpengaruh terhadap proses pencocokan. Sedangkan untuk perbedaan penulisan karakter tertentu atau khusus, baik jumlah atau letaknya dapat diabaikan pengaruhnya dalam proses pencocokan.

6. Kesimpulan

Pencocokan *string* berdasarkan kemiripan ucapan (*phonetic string matching*) yang merupakan bagian dari *inexact string matching* atau *fuzzy ztring matching* sangat bermanfaat untuk pencarian kata dengan kata masukan yang yang salah. Selain itu, juga bermanfaat jika dalam dokumen terdapat kesalahan penulisan, karena *phonetic string matching* mampu menunjukkan kecocokan selama ada kemiripan ucapan.

Hasil analasis kemampuan *phonetic string matching* algoritma *soundex*, *metaphone*, dan *caverphone*, menunjukkan bahwa kemampuan *phonetic string matching* yang baik ditunjukkan oleh algoritma caverphone. Dengan memperhatikan hasil analisis dan pengujian, juga dapat dilakukan perbaikan-perbaikan terhadap ketiga algoritma tersebut untuk meningkatkan kemampuan *phonetic string matching*.

Untuk penelitian lebih lanjut, sebaiknya dilakukan analisis ketiga algoritma dari segi kompleksitasnya. Selain itu, juga perlu dilakukan analisis pengaruh implementasi *phonetic string matching* pada sebuah *information retrieval system*.

Daftar Pustaka

[1] Binstock & Rex. 1995. *“Practical Algorithms for Programmers”*. Addison Wesley.

[2] Caversham Project. http://caversham.otago.ac.nz. Diakses tanggal 7 Desember 2004 pukul 09.00 WIB.

[3] Dictionary of Algorithms and Data Structures. National Institute of Standards and Technology. http://www.nist.gov/dads/. Diakses tanggal 3 Desember 2004 pukul 15.00 WIB.

[4] Dominic John Repici. 2004. *“How To : Understanding Classic Soundex Algorithms”*. http://www.creativyst.com/Doc/Articles/SoundEx1/SoundEx1.htm . Diakses tanggal 3 Desember 2004 pukul 15.00 WIB.

[5] Donal Knuth. 1973. *“The Art Of Computer Programming, vol. 3: Sorting And Searching”*. Addison Wesley.

[6] Jones Daniel. 1972. *“The Pronouncation of English”*. 4th edition. Cambridge : Great Britain at the University Press.

[7] Malmberg, Bertil. 1963. *Phonetics*. New York : Dover Publications.