Pencarian Nama yang Memiliki Kesamaan Fonetik Menggunakan Algoritme Kesamaan String

Julio Adisantoso,* Abdurrauf Rambe,* Indra Kaliana**

*) *Staf Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam*
**) *Mahasiswa Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam*

Abstrak

Kesamaan fonetik merupakan ukuran kesamaan yang sangat tergantung pada bahasa yang digunakan. Algoritme Soundex dan Phonix adalah metode yang terkenal untuk membandingkan kata dengan memperhatikan kesamaan fonetik. Dalam penerapan pada bahasa Indonesia, kedua metode tersebut mendapatkan masalah jika melakukan pencarian terhadap nama yang masih menggunakan ejaan lama. Penelitian ini bertujuan untuk menerapkan metode kesamaan string pada pencarian nama yang memiliki kesamaan fonetik dan membandingkan dua algoritme kesamaan string yaitu algoritme Uratani-Takeda dan algoritme Sekuensial.

Proses pencarian nama menggunakan metode kesamaan string ini terdiri dari dua langkah. Langkah pertama adalah membuat kombinasi kata yang sama fonetiknya dengan kata input. Kata-kata kombinasi yang terbentuk itu disebut varian. Langkah kedua adalah pencarian pada teks, katalog atau basis data nama menggunakan metode kesamaan string.

Pencarian nama menggunakan metode kesamaan string ini berhasil mengatasi masalah ejaan lama pada metode Soundex dan Phonix. Algoritme Uratani-Takeda memiliki kinerja yang cukup baik. Pada basis data dengan rata-rata panjang record 10 karakter waktu pencarian untuk input dengan panjang 8 karakter yaitu 3,454 detik dan 18,855 detik untuk input dengan panjang 3 karakter pada basis data dengan rata-rata panjang 250 karakter, lebih sedikit dari algoritme Sekuensial yaitu sebesar 3,629 detik untuk input 8 karakter dan 27,216 detik untuk basis data dengan rata-rata panjang data sebesar 250 karakter. Namun algoritme Uratani-Takeda membutuhkan ruang memori yang lebih besar dibandingkan dengan algoritme Sekuensial.

PENDAHULUAN

Latar Belakang

Setiap manusia memiliki nama diri untuk menunjukkan suatu identitas yang mudah diingat oleh orang lain. Nama diri kedudukannya berbeda dengan kata biasa. Nama diri, betapapun aneh ejaannya, tetap ditulis sesuai dengan ejaan yang digunakan oleh pemiliknya. Lain halnya dengan kata biasa, terutama kata-kata asing yang diserap menjadi kata dalam bahasa Indonesia, ejaannya disesuaikan dengan ejaan yang berlaku *(Mustakim, 1990)*.

Pencarian nama diri dalam basis data, katalog pengarang atau buku telepon merupakan suatu yang umum dalam sistem informasi. Jika diketahui dengan pasti ejaan nama diri itu, akan secara mudah ditemukan nama tersebut dalam daftar nama. Banyak metode yang dikembangkan untuk mencari sebuah *string* dalam sekumpulan *string* yang bisa dipakai untuk pencarian nama. Jika tidak diketahui ejaan yang tepat atau jika terjadi kesalahan mengetik saat mencari nama dalam daftar nama maka pencarian mungkin akan mengalami kegagalan, kecuali digunakan teknik pencarian tertentu.

Dalam bahasa Indonesia kemungkinan kesalahan dalam pencarian atau pembacaan nama diri akan terjadi karena ada nama yang menggunakan Ejaan Lama. Penulisan nama diri dalam bahasa Indonesia disesuaikan dengan aturan Ejaan yang Disempurnakan *(EYD)*, kecuali untuk nama diri yang telah disahkan sebelum EYD dan yang memiliki pertimbangan khusus, misalnya adat istiadat, hukum, kesejarahan, dan kesinambungan identitas pribadi *(Mustakim, 1990)*.

Penelitian untuk mencari nama yang memiliki kesamaan fonetik pernah dilakukan oleh Primasari *(1999)* menggunakan algoritme Soundex dan Phonix. Dalam penerapan pada bahasa Indonesia, kedua metode tersebut mendapatkan masalah jika melakukan pencarian terhadap nama yang masih menggunakan Ejaan Lama. Maka pada penelitian ini akan dicoba metode kesamaan *string* untuk mengatasi masalah pencarian nama yang masih menggunakan Ejaan Lama.

Tujuan

Tujuan dari penelitian ini yaitu :

1. Mempelajari dan menerapkan metode kesamaan *string* pada pencarian nama yang memiliki kesamaan fonetik untuk mengatasi masalah Ejaan Lama.
2. Membandingkan algoritme kesamaan *string* yang dikembangkan oleh Uratani-Takeda dengan algoritme Sekuensial.

TINJAUAN PUSTAKA

Menurut Pfeifer *et. al. (1996),* ukuran kesamaan non-linguistik dapat dibagi menjadi tiga kategori yang berbeda yaitu kesamaan string, kesamaan yang berhubungan dengan kesalahan pengetikan, dan kesamaan fonetik. Dari kategori pertama metode *n-grams* paling umum dipakai. Contoh dari kategori yang kedua yaitu metode Damerau-Levenstein-Metric, yang menghitung penyisipan, penghapusan, substitusi, dan transposisi yang dibutuhkan untuk mentransformasikan satu *string* ke *string* yang lain. Dalam dua kategori ini pembandingan kata-kata dilakukan tanpa memperdulikan jenis bahasa yang digunakan. Sedangkan untuk jenis kategori ketiga, ukuran kesamaan sangat tergantung pada bahasa yang digunakan. Algoritme Soundex dan Phonix adalah metode yang terkenal untuk membandingkan kata dengan memperhatikan kesamaan fonetik.

Arti kesamaan fonetik atau kesamaan bunyi yaitu adanya beberapa huruf dalam abjad yang memiliki bunyi pengucapan mirip antara huruf satu dengan huruf lainnya *(Primasari, 1999).* Sebagai contoh nama "Joko" dan "Djoko" merupakan dua nama yang memiliki kesamaan fonetik. Beberapa hal yang menyebabkan adanya kesamaan fonetik yaitu :

1. Kesalahan pendengaran, seperti pada nama Rina dan Rima.
2. Ingatan yang lemah, hal ini karena seseorang lupa akan nama yang akan dicari dan akibatnya dia melakukan pencarian dengan mencoba beberapa nama yang mirip bunyinya.
3. Kultur, seperti pada nama Ferri yang sering diucapkan dengan kata Perri.
4. Ejaan, seperti pada nama Kuncoro dan Koentjoro.
5. Kebiasaan.

Pada penelitian ini penyebab adanya kesamaan fonetik dibatasi pada faktor ejaan saja karena bahasa Indonesia mengalami berbagai perubahan ejaan dari semenjak penjajahan Belanda hingga muncul Ejaan Yang Disempurnakan.

Pengertian Ejaan

Pengertian ejaan secara umum berarti keseluruhan ketentuan yang mengatur pelambangan bunyi bahasa, termasuk pemisahan dan penggabungannya, yang dilengkapi pula dengan penggunaan tanda baca. Secara khusus ejaan berarti sebagai pelambangan bunyi-bunyi bahasa dengan huruf, baik berupa huruf demi huruf maupun huruf yang telah disusun menjadi kata, kelompok kata, atau kalimat *(Mustakim, 1990).*

Dalam suatu bahasa sistem ejaan lazimnya mempunyai tiga aspek, yaitu aspek fonologis yang menyangkut pelambangan fonem dengan huruf dan penyusunan abjad; aspek morfologis yang menyangkut pelambangan satuan-satuan morfemis, dan aspek sintaksis yang menyangkut pelambangan ujaran dengan tanda baca. Dengan demikian, ketentuan-ketentuan yang mengatur pelambangan fonem dengan huruf yang ada dalam bahasa Indonesia, serta pelafalan, pengakroniman, dan penyusunan abjad termasuk di dalam aspek fonologis. Ketentuan yang mengatur pembentukan kata dengan pengimbuhan, penggabungan kata, pemenggalan kata, penulisan kata, penyesuaian kosakata asing ke dalam bahasa Indonesia termasuk aspek morfologis. Di pihak lain, penulisan dan pelafalan frasa, klausa, serta kalimat termasuk aspek sintaksis.

Perkembangan Ejaan di Indonesia

Menurut Mustakim *(1990),* sejak masa penjajahan Belanda hingga masa kemerdekaan, ejaan dalam bahasa Indonesia mengalami bermacam perubahan. Perkembangan ejaan yang terjadi di Indonesia yaitu :

1. Ejaan Van Ophuysen
2. Ejaan Republik *(Ejaan Soewandi)*
3. Ejaan Melindo
4. Ejaan Baru *(Ejaan LBK)*
5. Ejaan Bahasa Indonesia Yang Disempurnakan *(EYD)*

Secara ringkas, huruf-huruf yang mengalami perubahan dari Ejaan Lama ke Ejaan Baru dapat dilihat pada ***Tabel 1.***

Tabel 1. *Perubahan huruf dari Ejaan Lama ke Ejaan Baru.*

Ejaan Lama	Ejaan Baru
tj	c
dj	j
j	Y
oe	u
ch	kh

Huruf-huruf dalam ***Tabel 1*** tersebut akan dipakai untuk membentuk varian dari input nama yang akan dicari.

Algoritme Kesamaan String

1. Algoritme Uratani-Takeda

Algoritme yang dipakai pada penelitian ini yaitu algoritme yang dikembangkan oleh Uratani dan Takeda *(1993)*.

Algoritme ini digunakan untuk mencari banyak input ke dalam *string* yang ide dasarnya mengadopsi dari algoritme pencarian *string* yang dikembangkan oleh Boyer dan Moore yakni pencarian *string* dilakukan dari kanan ke kiri, digabung dengan algoritme yang dikembangkan oleh Aho dan Corasick yaitu yang melakukan pencarian string untuk banyak *input.*

Untuk melihat cara kerja dari algoritme ini perhatikan contoh berikut :
Misalkan terdapat sekumpulan input yaitu (*trace, artist, smart*) yang akan dicari kedalam teks "*the-greatest-artist..*". Pertama kita menyamakan huruf terakhir dari semua input. Karena panjang input yg paling pendek adalah *5* maka kita mulai menyesuaikan dengan huruf ke-*5* dari teks dan *pointer* berada di huruf ke-*5* dari teks. Pada contoh di bawah ini, * menandakan sesuai, sedang x menunjukkan tidak sesuai.

```
 trace
artist
 smart
 the-greatest-artist..
     x
```

Karena huruf g tidak sesuai dengan huruf terakhir semua *input* dan tidak muncul di setiap kueri, kita menggeser *input-input* itu ke kanan sebanyak *5* huruf.

```
     trace
    artist
     smart
the-greatest-artist..
        x*
```

Pada kondisi ini, huruf e pada teks sesuai dengan huruf e pada input "*trace*". Karenanya kita menggeser *pointer* di teks ke kiri satu huruf. Karena tidak sesuai dan *string* te dari teks tidak muncul di semua input, kita geser lagi input-input itu sebanyak *5* huruf ke kanan *(atau menggeser pointer sebanyak 6 huruf).*

```
          trace
         artist
          smart
the-greatest-artist..
              *
```

Huruf r yang sekarang dibandingkan tidak sesuai dengan huruf terakhir dari semua kueri. Lalu kita meluruskan huruf r itu dengan huruf r yang ada di input "*smart*". Jadi kita menggeser *input-input* ke kanan sebanyak satu.

```
           trace
          artist
           smart
the-greatest-artist..
            x***
```

Kita dapat menyesuaikan huruf t, r, dan a dan gagal pada huruf -. Karena string art muncul pada input "*artist*" maka kita menggeser *input* tiga huruf ke kanan *(dan menggeser pointer ke kanan sebanyak enam).*

```
              trace
             artist
              smart
the-greatest-artist..
             ******
```

Pada saat ini, kita melihat bahwa semua huruf dari input "*artist*" sesuai dengan huruf yang ada di dalam teks.

2. Algoritme Sekuensial

Algoritme ini merupakan algoritme kesamaan *string* yang paling sederhana dari algoritme-algoritme kesamaan *string* yang ada. Algoritme Sekuensial melakukan pencarian *string input* terhadap teks dari kiri ke kanan dan dilakukan satu per satu karakter.

Misalkan $S=s_1s_2...S_n$ adalah teks dan $=p_1p_2...p_m$ adalah *string input* dimana $m \leq n$ maka algoritme Sekuensial adalah sebagai berikut :

1. *i := 1; j := 1;*
2. *if (i > n – m + 1) then*
3. *result := 0;*
4. *end program;*
5. *elseif* $(p_j = s_{i+j-1})$ *and* $(j = m)$ *then*
6. *result := 1;*

7. *end program;*
8. *elseif* $(p_j = s_{i+j-1})$ *and* $(j < m)$ *then*
9. *inc(j);*
10. *goto2;*
11. *else* $(p_j \neq s_{i+j-1})$ *then*
12. $j := 1$*; inc(i);*
13. *goto 2;*
14. *end;*

METODE PENELITIAN

Pengumpulan Data

Data yang digunakan yaitu data nama yang diambil dari buku petunjuk telepon kota Bogor tahun 2000 sebanyak 4370 buah. Diambil juga data alamat dan data nomor telepon dari nama yang bersangkutan.

Pembuatan Program

Program dibuat menggunakan perangkat lunak Microsoft Visual Basic 5.0 untuk aplikasinya dan Microsoft Acces 97 untuk basis datanya. Sedangkan perangkat keras yang digunakan yaitu komputer dengan *processor* AMD K5 PR-133, RAM 64 Mb dan kapasitas *harddisk* 1,2 Gb.

Alur dari program terdiri dari beberapa tahap:

1. *Input* nama.
2. Pembuatan varian dari *input* nama. Definisi varian yaitu kombinasi nama yang dapat dibentuk dengan merujuk pada huruf-huruf yang ada pada ***Tabel 1.***
3. Melakukan pencarian ke dalam basis data. Pencarian pada basis datanya sendiri dilakukan secara sekuensial yaitu dari *record* pertama hingga dari *record* terakhir. Sedang pencarian pada satu *record* digunakan algoritme Uratani-Takeda dan algoritme Sekuensial.
4. Tampilkan nama-nama yang ditemukan pada basis data dalam sebuah daftar.

Percobaan

Pada penelitian ini percobaan yang dilakukan yaitu :

1. Perhitungan rasio varian.

Percobaan ini dilakukan untuk mengetahui rasio varian yang ditemukan pada basis data dengan varian yang dibuat oleh program. *Input* nama yang digunakan yaitu Yanto, Fajar, Yahya, Cahjo, dan Yusuf.

2. *Pengukuran waktu*

Pada percobaan ini diamati waktu pencarian pada basis data dilihat dari dua faktor, yaitu perbedaan panjang input nama dan perbedaan panjang rata-rata nama pada basis data.

Untuk faktor perbedaan panjang *input,* nama-nama yang digunakan memiliki panjang 3-8 karakter *(masing-masing 10 nama)* yang nantinya akan diamati hubungan antara perubahan panjang *input* dengan waktu pencarian.

Sedang untuk faktor perbedaan panjang rata rata nama pada basis data dipakai basis data yang memiliki rata-rata panjang per *record* 10, 50, 100, 150, 200 dan 250 karakter. Kemudian diamati bagaimana hubungan antara perubahan panjang rata-rata basis data dengan perubahan waktu pencarian.

Sebagai perbandingan dilakukan juga pencarian dengan menggunakan algoritme Sekuensial. Kemudian diamati perbedaan waktu pencarian dari kedua algoritme tersebut.

HASIL DAN PEMBAHASAN

Rasio Varian

Pembentukan varian dilakukan oleh program dengan cara merubah huruf-huruf Ejaan Lama menjadi huruf-huruf Ejaan Baru atau sebaliknya dari huruf-huruf Ejaan Baru menjadi huruf-huruf Ejaan Lama sesuai dengan ***Tabel 1***. Jika ada lebih dari satu huruf, maka varian itu dibentuk sesuai dengan kombinasi dari huruf-huruf yang bisa dirubah. Misalkan nama "ayu" maka varian yang terbentuk yaitu "ayu", "ayoe", "aju", dan "ajoe".

Pada penelitian ini tidak diperhatikan kesesuaian varian khususnya terhadap varian yang kata utamanya terdapat huruf "j" dan "ch", tapi hanya kesamaan fonetik yang dipentingkan. Jika kata utamanya terdapat huruf "j", tidak ditekankan apakah "j" di sini merupakan huruf "j" yang masih Ejaan Lama yang lebih cenderung berubah ke huruf "y", atau "j" yang sudah berubah menjadi Ejaan Baru *(jika dirubah ke Ejaan Lama maka menjadi "dj").* Sama halnya dengan "ch" apakah lebih cenderung untuk berubah menjadi "tj" seperti pada nama "chandra" atau lebih ke huruf "kh" seperti pada kata "achmad".

Hasil percobaan penghitungan rasio varian ini dapat dilihat pada ***Tabel 2.*** Dari tabel ini diketahui rasio varian yang ditemukan pada basis data beragam dari 67% hingga 100%.

Tabel 2. *Rasio antara varian yang terbentuk oleh program dengan yang didapatkan dari basis data.*

Input Nama	Varian				Rasio
	Oleh program	Jumlah	Dari basis data	Jumlah	
yanto	yanto, janto	2	yanto, janto	2	100%
fajar	fajar, fadjar, fayar	3	fajar, fadjar	2	67%
yahya	yahya, yahja, jahya, jahja	4	yahya, jahya, jahja	3	75%
cahjo	cahjo, cahyo, cahdjo, tjahjo, tjahyo, tjahdjo	6	cahjo, cahyo, tjahjo, tjahyo	4	67%
yusuf	yusuf, yusoef, yoesuf, yoesoef, jusuf, jusoef, joesuf, joesoef	8	yusuf, yusoef, yoesuf, yoesoef, jusuf, jusoef, joesuf, joesoef	8	100%

Besarnya nilai rasio varian ini dipengaruhi beberapa hal. Yang pertama yaitu karena basis data yang dipakai hanya mengambil sebagian kecil dari nama-nama yang ada di buku telepon. Jika memakai seluruh data nama yang terdapat pada buku telepon, kemungkinan rasio varian akan mencapai 100% besar. Seperti pada input nama "yusuf" yang memiliki rasio varian sebesar 100%. Hal ini karena semua varian dari nama "yusuf" dimasukkan ke dalam basis data sebelum percobaan dilakukan.

Hal berikutnya yaitu tidak diperhatikannya arah perubahan huruf "j". Sebagai contoh, pada nama "fajar", huruf "j" di sini cenderung untuk berubah menjadi huruf "j" atau "dj" daripada berubah ke huruf "y". Hal ini dapat dilihat dari hasil yang didapatkan dari basis data hanya terdapat nama "fajar" dan "fadjar". Sedangkan untuk nama "cahjo", huruf "j" di sini lebih diucapkan menjadi huruf "y" dari pada menjadi huruf "j" atau "dj" sehingga varian yang didapatkan yaitu "cahjo", "cahyo", "tjahjo", dan "tjahyo".

Tidak diketahui apakah varian yang terbentuk merupakan sebuah nama depan, nama tengah, atau nama belakang yang dipisahkan oleh spasi ataukah merupakan potongan dari sebuah nama juga mempengaruhi besar kecilnya rasio varian. Contoh varian "jahya" yang merupakan potongan dari nama "tjahya" dan varian "janto" yang merupakan potongan dari kata "harjanto" ikut terambil. Akibatnya nilai rasio akan bertambah walau sebenarnya hasil yang didapatkan belum tentu relevan dengan nama yang diinginkan.

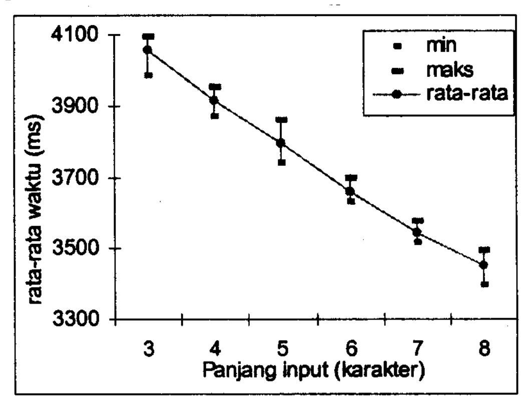
Jika kita memperhatikan hal-hal tersebut, maka bisa ditentukan tingkat relevansi dari varian yang terbentuk dan nantinya akan berpengaruh pada *output* yang dihasilkan.

Pengukuran Waktu

Pengukuran waktu pada penelitian ini hanya dilakukan saat melakukan pencarian pada basis data. Pada percobaan, pengukuran waktu dicatat hingga seperseribu detik atau mili sekon dan dilakukan sebanyak sepuluh kali ulangan untuk tiap *input* nama pada tiap basis data yang dipakai, kemudian dirata-ratakan.

1. Hubungan Waktu dan Panjang Input Nama

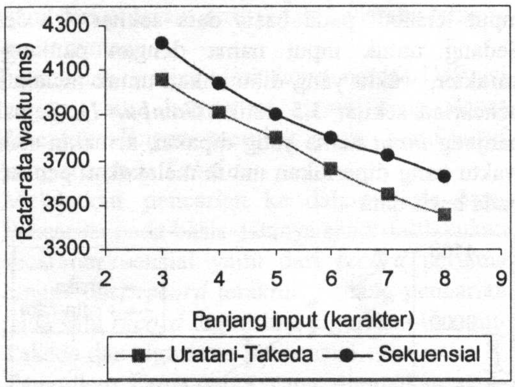
Untuk *input* nama yang memiliki panjang *3* karakter, waktu yang dibutuhkan untuk mencari input tersebut pada basis data sekitar 4,1 detik. Sedang untuk input nama dengan panjang *8* karakter, waktu yang dibutuhkan untuk melakukan pencarian sekitar 3,5 detik ***(Gambar 1).*** Semakin panjang *input* nama yang dipakai, semakin sedikit waktu yang diperlukan untuk melakukan pencarian pada basis data.



Gambar 1 *Waktu pencarian pada satu basis data dengan berbagai input.*

Semakin cepatnya waktu pencarian seiring dengan pertambahan panjang *input* merupakan kelebihan dari algoritme Uratani-Takeda ini. Hal ini karena proses penyesuaian string yang dimulai dari kanan pada posisi karakter ke *minlen* dari teks. Jika ada ketidaksesuaian karakter maka posisi *pointer* digeser ke kanan sebanyak *minlen* karakter. *Minlen* yaitu panjang *input* terpendek jika terdapat lebih dari satu input. Jika hanya satu *input* maka *minlen* adalah panjang *input* tersebut. Untuk *input* nama yang hanya 3 karakter panjangnya, maka akan ada pergeseran *pointer* ke kanan sebanyak 3 karakter. Jika *minlen* sebesar 4, maka *pointer* bergeser sebanyak 4 karakter. Untuk *input* nama yang memiliki nilai *minlen* sebesar 8, maka akan terjadi pergeseran karakter sebanyak 8. Semakin besar panjang *input,* semakin besar nilai *minlen* yang dimiliki. Akibatnya pergeseran *pointer* akan semakin jauh, yang berakibat waktu pencarian akan semakin pendek.

Jika dibandingkan terhadap algoritme Sekuensial dengan input dan basis data yang digunakan sama, terlihat bahwa algoritme Uratani-Takeda memiliki waktu pencarian yang lebih cepat untuk semua *input* yang dipakai ***(Gambar 2)***. Untuk *input* nama dengan panjang 3 karakter waktu pencarian sebesar 4,1 detik, sedang untuk input nama dengan panjang *8* karakter waktu yang dibutuhkan untuk melakukan pencarian dalam basis data sebesar 3,5 detik.



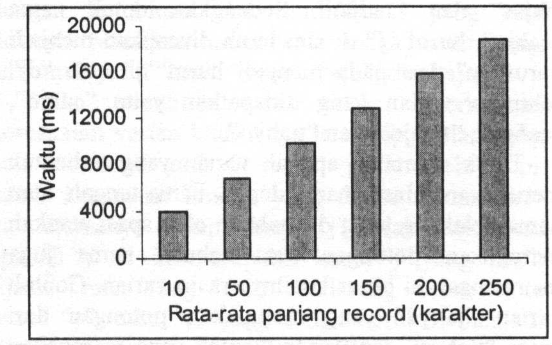
Gambar 2 *Grafik perbandingan waktu pencarian antara algoritme Uratani-Takeda dan algoritme Sekuensial pada beberapa input.*

Walaupun pada algoritme Sekuensial semakin panjang *input* semakin kecil waktu pencarian, tetapi masih lebih besar dibandingkan waktu pencarian menggunakan algoritme Uratani-Takeda. Ini disebabkan oleh perbedaan proses pencarian pada masing-masing algoritme. Jika pada algoritme Uratani-Takeda terjadi pergeseran *pointer* sebesar *minlen* dengan besarnya *minlen* tergantung pada panjang input nama. Pada algoritme Sekuensial pergesaran *pointer* yang terjadi hanya sebesar satu karakter untuk semua panjang *input.* Sehingga jumlah perbandingan semakin banyak dan akibatnya waktu pencarian menjadi lebih besar.

2. Hubungan Waktu dan Panjang Rata-rata Data

Pada percobaan ini akan diamati pengaruh perubahan panjang rata-rata nama per *record* pada basis data terhadap waktu pencarian. *Input* nama yang digunakan yaitu *input* nama dengan panjang *3* karakter. Basis data yang digunakan ada *6* file, yaitu Telp.mdb yang berisikan data nama-nama yang asli dari sumber dan *5* file lainnya *(TelpP50.mdb, TelpP100.mdb, TelpP150.mdb, TelpP200.mdb, TelpP250)* yang berisi data nama-nama yang ditambahkan karakter secara acak di depan nama yang asli sehingga panjangnya menjadi 50-250 karakter. Semua file data masing-masing berisi 4370 *record.* Hasil percobaan ditunjukkan oleh histogram pada ***Gambar 3.***

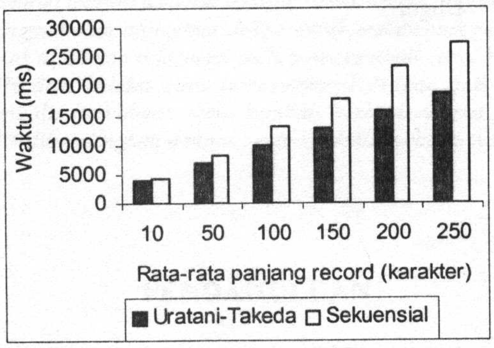
Gambar *3* menunjukkan ada peningkatan waktu pencarian dengan bertambahnya rata-rata panjang pada basis data. Untuk basis data dengan rata-rata panjang nama sebesar *10* karakter waktu yang dibutuhkan untuk mencari *input* nama dengan panjang *3* karakter sebesar 4,1 detik. Dengan menggunakan *input* nama yang sama, waktu pencarian pada basis data yang rata-rata panjang datanya 250 karakter yaitu sekitar 18,9 detik. Semakin besar rata-rata panjang data maka waktu pencarian akan semakin lama.



Gambar 3. *Waktu pencarian pada beberapa basis data*

Kejadian seperti ini adalah hal yang umum terjadi pada temu kembali informasi yang berbasis teks. Semakin besar ukuran teks yang pada basis data, maka semakin lama waktu yang diperlukan, pencarian akan semakin tidak efisien *(Salton, 1989).* Semakin banyak jumlah karakter pada basis data, maka akan mengakibatkan bertambahnya jumlah perbandingan saat melakukan pencarian.

Pada algoritme Sekuensial, waktu pencarian untuk *input* nama sepanjang 3 karakter lebih besar daripada waktu yang dibutuhkan oleh algoritme Uratani-Takeda. Pada Gambar 4 dapat dilihat ada perbedaan antara algoritme Uratani-Takeda dan algoritme Sekuensial. Pada basis data yang memiliki rata-rata panjang *10* karakter, perbedaan waktu antara algoritme Uratani-Takeda dengan algoritme Sekuensial tidak terlalu jauh yaitu sekitar 0,17 detik. Tetapi pada basis data dengan panjang rata-rata 250 karakter perbedaan waktu antara kedua algoritme itu lebih dari 8,4 detik. Dari sini dapat terlihat kelebihan dari algoritme Uratani-Takeda dibandingkan dengan algoritme Sekuensial.



Gambar 4. *Histogram perbandingan waktu pencarian antara algoritme Uratani-Takeda dan algoritme Sekuensial pada beberapa basis data.*

Analisis Algoritme

Telah diketahui sebelumnya bahwa pada algoritme Uratani-Takeda perpindahan posisi *pointer* sebanyak *minlen* dimana *minlen* merupakan panjang *input* terpendek. Pada kasus terburuk, yakni semua karakter *input* tidak sama dengan semua karakter teks yang dicari, maka pada setiap tahap hanya akan ada satu perbandingan dan pergeseran sebesar *minlen*. Jika panjang total teks yang dicari adalah *N* maka total perbandingan yang terjadi pada kasus terburuk yaitu sebanyak $\lfloor N/minlen \rfloor$ perbandingan.

Pada algoritme Sekuensial, untuk mencari pada teks yang sama dengan panjang N dengan *input* yang dipakai memiliki panjang sebesar *m*, pergeseran *pointer* hanya sebanyak satu karakter sehingga pada kasus terburuk banyaknya perbandingan yang terjadi yaitu sebesar N-m+1 perbandingan.

Pada algoritme Uratani-Takeda, terdapat fungsi *goto*, *output*, dan *failure*. Fungsi *goto* dan fungsi failure merupakan *array* dua dimensi berukuran $|\Sigma| \times \partial$, dengan $|\Sigma|$ adalah jumlah anggota dalam himpunan karakter dan ∂ yaitu jumlah *state*. Pada penelitian ini yang digunakan hanya karakter a-z dan spasi. Jumlah *state* maksimal sama dengan jumlah panjang semua *input* nama ditambah satu. Untuk fungsi *output* merupakan sebuah vektor berukuran *l*. Jadi algoritme Uratani-Takeda memerlukan ruang memori sebanyak $(2 \times |\Sigma| \times \partial) + 1$ sedangkan algoritme Sekuensial tidak ada memori tambahan. Algoritme ini hanya memerlukan dua buah variabel yang berfungsi sebagai *pointer*. Karena itu ruang memori yang diperlukan algoritme Sekuensial lebih sedikit.

PENUTUP

Pencarian nama menggunakan metode kesamaan *string* ini dapat mengatasi masalah Ejaan Lama pada metode Soundex dan Phonix.

Algoritme Uratani-Takeda memiliki kinerja yang lebih baik dibanding algoritme Sekuensial. Rata-rata waktu yang dibutuhkan oleh algoritme Uratani yaitu sebesar 3,454 detik untuk *input* dengan panjang 8 karakter pada basis data dengan rata-rata panjang data 10 karakter dan 18,855 detik untuk *input* dengan panjang 3 karakter pada basis data dengan rata-rata panjang 250 karakter, lebih sedikit dari algoritme Sekuensial yaitu sebesar 3,629 detik untuk *input* 8 karakter dan 27,216 detik untuk basis data dengan rata-rata panjang data sebesar 250 karakter. Tetapi ruang memori yang dibutuhkan oleh algoritme Uratani-Takeda jauh lebih banyak dibanding algoritme Sekuensial.

Metode ini masih dibatasi pada faktor ejaan khususnya pada nama yang masih memiliki Ejaan Lama. Karena itu masih dapat dikembangkan lagi tidak hanya untuk faktor ejaan.

Faktor-faktor yang lain seperti nama yang merupakan serapan dari bahasa asing *(Arab, Inggris, Cina)* atau faktor-faktor penyebab kesamaan fonetik lainnya, seperti kesalahan pendengaran, ingatan yang lemah, kultur dan faktor kebiasaan perlu diperhatikan. Selain itu juga faktor kesesuaian dari varian yang terbentuk.

DAFTAR PUSTAKA

Boyer, R. S. & Moore, J. S. 1977. A Fast String Searching Algorithm. *Comm. ACM* 20:762-772.

Hall, P. A. V. & Dowling, G. R. 1980. Approximate String Matching. *ACM Comput. Surveys*. 12:381-402.

Mustakim. 1990. *Tanya Jawab Ejaan Bahasa Indonesia Untuk Umum.* PT. Gramedia Pustaka Utama, Jakarta.

Pfeifer, U., T. Poersch & N. Fuhr. 1996. Retrieval Effectivenes of Proper Name Search Methods. *Information Processing & Management* 32:667-679.

Primasari, D. 1999. Temu Kembali Nama Menggunakan Metode Kesamaan Fonetik. Skripsi. Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Pertanian Bogor.

Salton, G. 1989. *Automatic Text Processing: The Information, Analysis,and Retrieval of Information by Computer.* Addison-Wesley Publishing Company. Canada.

Uratani, N. & Takeda, M. 1993. A Fast String Searching Algoritm for Multiple Patterns. *J. inform. Sist. Mgmt.* 29:775-791.

Wahyudin, A. 1999. Algoritme Trigram Untuk Mengoreksi Ejaan. Skripsi. Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Pertanian Bogor.