Artikel ini telah dipresentasikan dalam *Innovative and Creative Information Technology Conference* (ICITech) dengan tema *"E-Transaction and Power Play"* yang diselenggarakan oleh Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana – Salatiga, pada 24 November 2016.

PENGKOREKSIAN DAN *SUGGESTION WORD* PADA *KEYWORD* MENGGUNAKAN ALGORITMA *JARO WINKLER*

Kristien Margi S[1], Agus T[2]
[1,2]Teknik Informatika, Fakultas Teknologi dan Desain, Universitas Bunda Mulia
Jalan Lodan Raya No. 2, Jakarta 14430
Email : [1)]ksuryaningrum@bundamulia.ac.id

Abstrak

Keyword merupakan kata kunci yang digunakan untuk mempermudah dalam pencarian sumber bahan pustaka. Proses pencarian melalui keyword untuk menemukan judul buku atau bahan pustaka telah menjadi kebutuhan pokok dalam suatu sistem. Namun terkadang saat pengetikan keyword, masih banyak terjadi kesalahan penulisan atau typo. Tujuan dari penelitian ini adalah membuat suatu aplikasi dalam bentuk prototype yang memeberikan koreksi atau Suggestion word pada keyword sebuah bahan pustaka.Metode Jaro-winkler merupakan metode string matching yang berguna untuk mencocokan dua buah string, dengan masing-masing rumus algoritma dan juga proses yang berbeda. Pengambilan data dilakukan dengan cara mengambil data buku dari perpustakaan kemudian digunakan sebagai sample dalam proses pencarian keyword. Hasil dari penelitian ini adalah sistem yang mampu memberikan koreksi dan Suggestion word jika dalam pencarian data buku terjadi kesalahan pengetikan. Diharapkan dengan dibangunnya sistem ini dapat mempermudah pengguna perpustakaan dalam mencari buku.

Keywords*: Jaro Winkler, Suggestion word, Keyword*

1. PENDAHULUAN

Pengolahan bahan pustaka merupakan salah satu inti dari tugas perpustakaan. Bahan pustaka yang masuk ke perpustakaan wajib diolah dengan baik agar proses temu kembali informasi nantinya berjalan lancar dan mewujudkan tertib administrasi. Dalam pelaksanaannya, proses pengolahan bahan pustaka ini dapat berbeda-beda urutan kegiatan atau alur prosesnya antara perpustakaan satu dengan yang lainnya. Hal ini mungkin disebabkan oleh adanya perbedaan budaya kerja, sumber daya manusia, dan sarana prasarana dalam proses pengolahan.

Bahan pustaka atau buku yang dicari, biasanya melalui proses *search engine* pada suatu aplikasi pencarian buku. *Search engine* merupakan aktivitas yang biasa dilakukan oleh user. Pada penerapannya, pencarian kata bertujuan untuk menemukan kata kunci yang dicari di dalam suatu teks untuk mencari suatu informasi yang relevan dengan kata kunci, seperti mencari kata yang sama atau mengecek kesamaan antara dua buah teks

. Pentingnya sistem pencarian kata ini dapat telihat jelas di dalam sebuah perpustakaan. Namun terkadang dalam pengetikan pencarian bahan pustaka, *user* bisa saja salah memasukkan kata yang dimaksud. Hal ini dikarenakan salah pengetikan (*typo*) atau kata yang dimaksud tidak sesuai dengan yang ada di *database*. Untuk itu, diperlukan suatu sistem yang dapat membantu *user* untuk melakukan pengkoreksian *keyword* dan memberikan *Suggestion word* yang tidak sesuai dengan *keyword* yang dimaksud. Hal ini, untuk mempermudah *user* supaya mengarahkan dalam proses pencarian daftar pustaka, melalui *keyword* utama.

Konsep ini diterapkan seperti pada penelitian sebelumnya yang sudah dilakukan mengenai menghitung tingkat kesamaan dokumen berbasis web menggunakan pemecahan metode untuk menghitung tingkat kesamaan dokumen dengan menggunakan Algoritma *jaro-winkler distance* [1]. Dari konsep tersebut, metode *Jaro Winkler Distance* akan diterapkan untuk memecahkan permasalahan pengkoreksian kesalahan pengetikan, karena salah satu ciri utama metode ini adalah menemukan jumlah karakter yang sama persis, sehingga dapat membantu dalam pengkoreksian karena kesalahan pengetikan dan kemudian memberikan suggestion word untuk keyword yang diinputkan

2. METODOLOGI

Dalam studi penelitian yang dilakukan sebelumnya telah dihasilkan sebuah aplikasi Untuk dapat mengukur tingkat kesamaan dokumen dengan cepat, maka diperlukan alat bantu untuk dapat menghitung tingkat kesamaan antar dokumen berbasis *web*. Penelitian ini bertujuan mengimplementasikan salah satu metode *approximate string matching*, yakni *Jaro Winkler* dalam mengatasi permasalahan tersebut. Pembahasan mengenai proses penghitungan jarak antar *string*, proses kerja mesin pencari berbasis *web* yang meliputi proses *crawling*, *indexing* sampai dengan menampilkan hasil pencarian beserta saran kata kunci adalah fokus pada penelitian ini. Pengujian terhadap aplikasi ini dengan menggunakan data abstraksi penulisan atau penelitian ilmiah jurusan Sistem Informasi Universitas Gunadarma sebanyak 50 abstraksi [1]

Pada penelitian sebelumnya yang lain, menghasilkan sebuah aplikasi untuk menghitung tingkat kesamaan dokumen teks berbahasa Indonesia berbasis desktop, dengan menerapkan algoritma Jaro-Winkler distance. Tujuan dari penerapan algoritma ini adalah membandingkan kesamaan antar dokumen teks berbahasa Indonesia, sehingga dapat ditentukan sebuah dokumen tersebut plagiat atau tidak. Dengan penerapan sebuah program, masalah tersebut dapat diatasi. Dalam lingkup yang kecil, misalnya kegiatan mahasiswa, penerapan ini akan sangat berguna, misalnya pada pengoreksian karya tulis, skripsi dan sebagainya [2].

Pada penelitian yang melakukan penelitian yang sama, menghasilkan sebuah aplikasi unruk menampilkan saran perbaikan kesalahan pengetikan dokumen berbahasa Indonesia. Algoritma yang digunakan untuk memberikan saran perbaikan adalah *Jaro Winkler* yang dapat menghitung keterkaitan antar *string* dan menghitung jumlah keterbedaan antar dua *string* [3]

2.1 LANDASAN TEORI

2.1.1 *JARO WINKLER*

Jaro-Winkler merupakan varian dari Jaro Distance metrik yaitu sebuah algoritma untuk mengukur kesamaan antara dua buah *string*, biasanya algoritma ini digunakan di dalam pendeteksian duplikat. Semakin tinggi nilai Jaro-Winkler untuk dua *string*, maka semakin tinggi presentase kemiripan kedua buah *string* tersebut. [1]

Langkah dasar untuk menghitung algoritma Jaro antara dua *string* s1 dan s2 adalah sebagai berikut:

a. Menghitung panjang *string* s1 dan s2.

b. Menemukan jumlah karakter yang "sama persis"(m) dalam dua *string* yang dibandingkan.
c. Menghitung jumlah transposisi kedua buah *string*.

Algoritma Jaro mendefinisikan "karakter yang sama" sebagai karakter pada kedua *string* yang sama dan memenuhi ketentuan jarak teoritis [1].

$$\left\lfloor \frac{\max(|s1|,|s2|)}{2} \right\rfloor - 1 \quad \text{Rumus 1 Karakter Sama}$$

$$d_j = \frac{1}{3}\left(\frac{m}{|s1|} + \frac{m}{|s2|} + \frac{m-t}{m}\right) \quad \text{Rumus 2 } \textit{Jaro Distance}$$

Pengukuran algortima *Jaro-Winkler* adalah ekstensi dari algoritma *Jaro Distance.*

$$d_w = d_j + (\boldsymbol{l} * \boldsymbol{p}(1 - d_j)) \quad \text{Rumus 3 } \textit{Jaro-Winkler Distance}$$

Ekstensi ini dibuat berdasarkan observasi Winkler bahwa kesalahan pengetikan biasanya muncul di tengah pengetikan atau di akhir kata, sangat jarang terjadi kesalahan pengetikan di awal kata

2.2.2 PREPROCESSING

Preprocessing adalah tahapan mengubah suatu dokumen kedalam format yang sesuai agar dapat diproses lebih lanjut. Pada penelitian ini, tahapan *preprocessing* yang dilakukan adalah *tokenizing, case folding,* dan *number removal*. Berikut penjelasan masing-masing tahapan *preprocessing* [1].

1. *Tokenizing*
   Tokenizing adalah proses untuk membagi teks yang dapat berupa kalimat, paragraf atau dokumen, menjadi token-token/bagian-bagian tertentu. Sebagai contoh, tokenisasi dari kalimat "Aku baru saja makan bakso pedas" menghasilkan enam token, yakni: "Aku", "baru", "saja", "makan", "bakso", "pedas". Biasanya, yang menjadi acuan pemisah antar token adalah spasi dan tanda baca.
2. *Case Folding*
   Case folding adalah tapahapan dimana semua huruf dalam dokumen akan dirubah menjadi huruf kecil, untuk mempermudah pencocokan *string* nantinya yang bersifat *case sensitive*

.

2.2.3 *Text Mining*

Text Mining adalah sebuah penerapan yang berasal dari *Information Retrieval* (IR) dan natural language processing (NLP). Definisi *Text Mining* secara sempit hanya berupa metode yang dapat menemukan informasi baru yang tidak jelas atau mudah diketahui dari sebuah kumpulan dokumen. Sedangkan secara lebih luas, *Text Mining* mencakup teknik text-processing yang lebih umum, seperti pencarian, pengambilan intisari, dan pengkategorian.

Permasalahan yang dihadapi pada *Text Mining* adalah jumlah data yang besar, dimensi yang tinggi, data dan struktur yang terus berubah, serta data noise. Sehingga sumber data yang digunakan pada *Text Mining* adalah kumpulan teks

yang memiliki bentuk yang tidak terstruktur atau setidaknya semi terstruktur. Tujuan dari *Text Mining* adalah untuk mendapatkan informasi yang berguna dari sekumpulan dokumen dalam bentuk teks [4][5].

2.2.4 *Information Retrieval*

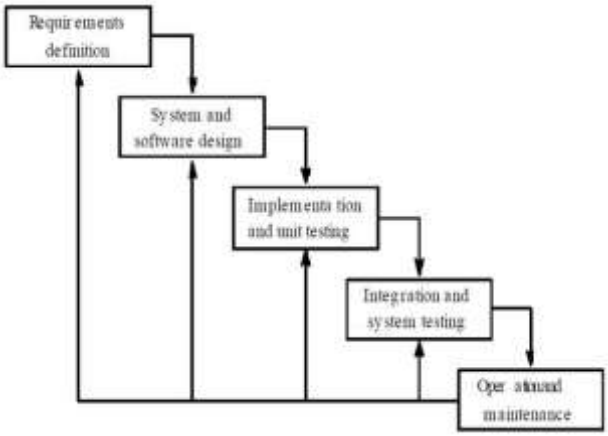
Definisi *Information Retrieval* (IR) adalah bagaimana menemukan suatu dokumen dari dokumen-dokumen tidak terstruktur yang memberikan informasi yang dibutuhkan dari koleksi dokumen yang sangat besar yang tersimpan dalam komputer. (Manning, 2008). Tujuan dari sistem IR adalah untuk memenuhi kebutuhan informasi pengguna dengan meretrieve semua dokumen yang mungkin relevan, pada waktu yang sama me-retrieve sesedikit mungkin dokumen yang tidak relevan. Sistem IR yang baik memungkinkan pengguna menentukan secara cepat dan akurat apakah isi dari dokumen yang diterima memenuhi kebutuhannya. Tujuan yang harus dipenuhi adalah bagaimana menyusun dokumen yang telah didapatkan tersebut ditampilkan terurut dari dukumen yang memiliki tingkat relevansi tinggi ke tingkat relevansi yang lebih rendah. Penyusunan dokumen tersebut disebut sebagai perangkingan dokumen [4].

2.2 METODE PENELITIAN

Metode yang digunakan untuk mendapatkan data dan informasi yang relevan dengan permasalahan yang dibahas, yaitu menggunakan sistem *waterfall*.

Metode perancangan yang digunakan pada pembuatan aplikasi ini yaitu model *waterfall*. Metode *waterfall* adalah suatu proses pembuatan situs *web* secara terstruktur dan berurutan dimulai dari penentuan masalah, analisa kebutuhan, perancangan implementasi, integrasi, uji coba sistem, penempatan situs *web* dan pemeliharaan. Pembuatan situs *web* dengan metode ini sangat cocok dilakukan pada situs *web* berskala besar karena menyangkut manajemen dan sistem yang rumit. Metode ini membutuhkan pendekatan sistematis dan sekuensial dalam pengembangan perangkat lunak dan biasanya disebut juga dengan *Classic Life Cycle*, dimulai dari tingkat sistem dan kemajuan melalui *analysis, design, coding, testing* dan *maintenance*.

Implementasi dari perancangan sistem yang dibuat, akan dibangun dan diterapkan ke dalam bahasa pemrograman PHP dan My SQL. Bahasa pemrograman PHP, merupakan bahasa pemrograman *web server-side* yang bersifat *open source*, sehingga sistem yang dibangun akan lebih dinamis. *Database* ini dibuat untuk keperluan sistem *database* yang cepat, handal dan mudah digunakan. Secara detail, alur model *waterfall* yang merupakan model klasik akan digambarkan pada Gambar 1.

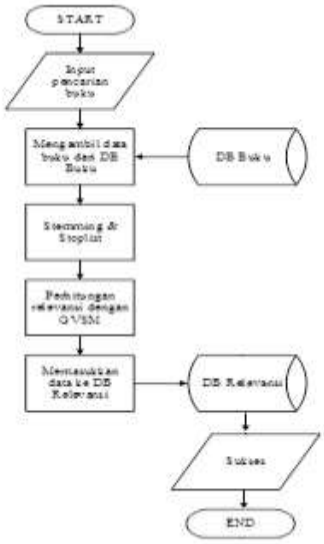


Gambar 1 Alur Model *Waterfall* [6]

1. *Requirements/ Analysis*
   Fase *Requirements/ Analysis* dikerjakan agar menghasilkan desain sistem yang lengkap. *Requirements* adalah tahapan untuk menentukan kebutuhan data yang akan digunakan untuk membangun sistem. Data-data kemudian secara lengkap dianalisa untuk dijadikan metode dalam pengembangan sistem informasi beserta kebutuhan *database* yang harus dipenuhi oleh program yang akan dibuat.
2. *Design*
   Proses desain menterjemahkan kebutuhan pengguna dalam sebuah dokumen aplikasi yang dapat diperkirakan kualitasnya sebelum proses *coding* dimulai. Tahapan *design menentukan* bagaimana cara menyelesaikan masalah dengan mengedepankan fokus pada rancangan fisikal seperti struktur data, tampilan layar, *database*. Dari hasil analisis analisis terhadap data dan metode, perancangan aplikasi dilakukan dengan menggunakan *flowchart, data flow diagram* (DFD), arsitektur sistem, *flowchart* dan *entity relationship diagram* (ERD). Proses perancangan aplikasi dilakukan untuk mendapatkan gambaran sebelum melakukan proses implementasi
3. *Implementation / Code*
   Tahap implementasi adalah tahap dimana hasil desain *software* diterjemahkan ke dalam bahasa yang dapat dimengerti oleh komputer. Sehingga pada tahap ini perancangan akan dibangun ke dalam coding melalui PHP dan My SQL.
4. *Testing*
   Untuk tahap testing, setelah dibuat program aplikasi kemudian diuji untuk mengetahui apakah sudah dapat berfungsi seperti yang diinginkan. Pengujian ini menggunakan teknik *white box* atau *black box*.
5. *Maintenance*
   Tujuan dari tahapan ini adalah melakukan dukungan dan pemeliharaan terhadap implementasi sistem agar tetap berf
   ungsi pada tingkat yang lebih tinggi.

2.4 *FLOWCHART* SISTEM

Bagan alir (*flowchart*) digunakan terutama untuk alat bantu komunikasi dan untuk dokumentasi. *Flowchart* sistem ditunjukkan pada Gambar 1. [6].



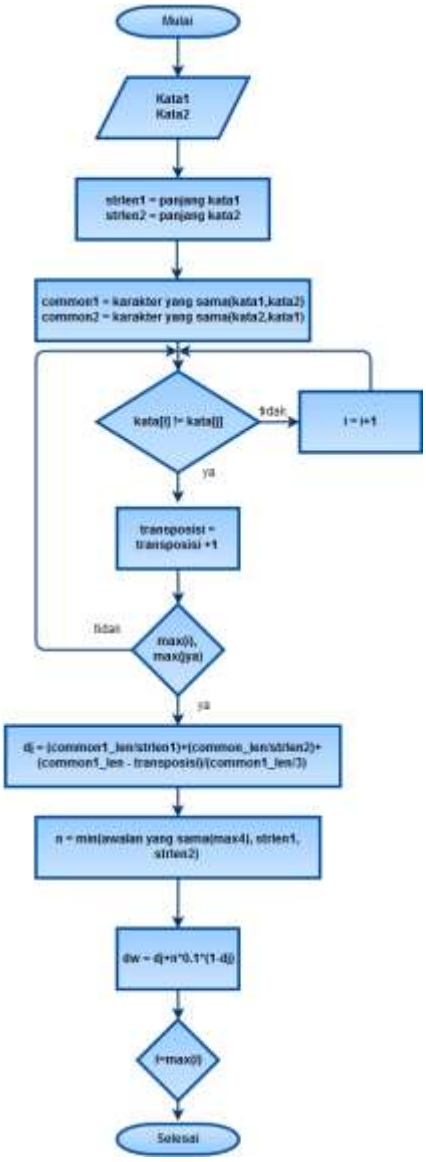
Gambar 1 *Flowchart* sistem

Sedangkan *flowchart* algoritma *Jaro Winkler* ditunjukkan pada Gambar 2. Gambar 2 adalah *flowchart* untuk pencarian nilai *Jaro-Winkler* antara *keyword* dengan data buku pada *database*, dan merupakan cara kerja algoritma *Jaro-Winkler*.

Langkah-langkahnya sebagai berikut: [5]

1. Input kata1 dan input kata2(berasal dari *database* buku).
2. Tentukan panjang kata1=strlen1 dan panjang kata2 =strlen2.
3. Hitung *distance maximal* untuk perhitungan *Jaro distance*, dengan cara max(strlen1, strlen2) /2-1.
4. Ambil karakter yang sama dengan parameter:
   - Common1 = karakter yang sama (kata1, kata2), kata 1 sebagai acuan perbandingan.
   - Common2 = karakter yang sama(kata2, kata1), kata 2 sebagai acuan perbandingan.
5. Periksa apakah karakter pada kata1 sama dengan karakter pada kata2, jika sama maka periksa karakter selanjutnya(i+1), jika tidak sama maka transposisi = transposisi + 1. Ulangi sampai karakter pada kata1 dan kata2 habis diperbandingkan.
6. Hitung nilai *Jaro* nya yang akan menjadi acuan perhitungan nilai *Jaro-Winkler*.
7. Hitung awalan yang sama yang terdapat pada kedua karakter. Maksimal awalan yang sama = 4 karakter, lalu cari nilai minimal antara (awalan yang sama, strlen1, strlen2). Bagian inilah yang merupakan ekstensi dari algoritma *Jaro-Winkler*. Winkler meyakini bahwa kesalahan pengetikan biasa terjadi di tengah atau di akhir kalimat, rumus inilah yang menjadi representasi teori Winkler, dan juga merupakan keunggulan algoritma *Jaro-Winkler* dibandingkan dengan algoritma *Jaro*.

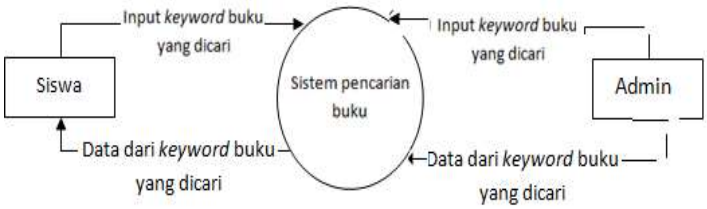
8. Langkah terakhir adalah menghitung nilai *Jaro-Winkler*.



Gambar 2 *Flowchart Jaro Winkler*

2.5 DATA FLOW DIAGRAM

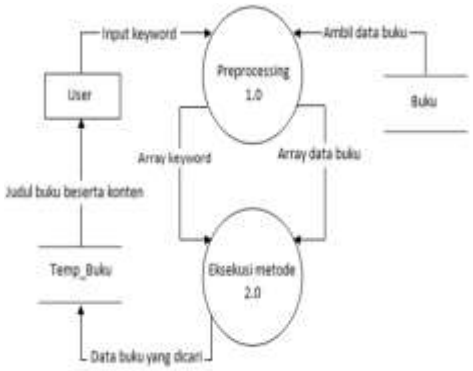
Data Flow Diagram (DFD) adalah suatu diagram yang menggunakan notasi-notasi untuk menggambarkan arus dari data pada suatu sistem, yang penggunaannya sangat membantu untuk memahami sistem secara logika, tersruktur dan jelas [4][5]. DFD sangat mirip dengan *Flowchart*. *Diagram context* aplikasi ini dapat digambarkan pada Gambar 3.



Gambar 3 Diagram Context

Gambar 3 menunjukkan terdapat dua entitas yang akan berinteraksi dengan sistem yang akan dibangun, yaitu entitas *user dan Admin*. *User* berperan sebagai entitas luar yang akan berinteraksi langsung dengan system, sedangkan Admin berperan sebagai entitas luar yang akan mengolah data

Hasil penjabaran dari diagram konteks adalah diagram *level* 0. Pada diagram *level* ini, proses dijabarakan lagi sesuai dengan proses-proses utamanya. DFD *level* 0 ini, ditunjukkan pada Gambar 4



Gambar 4 DFD *Level* 0

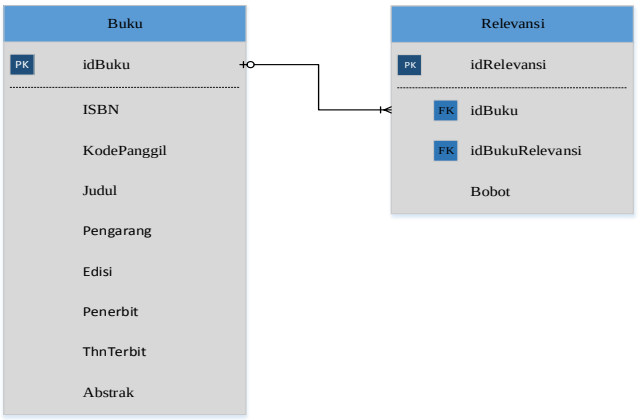
Pada diagram *level* 0 yang terbentuk, terdapat 2 proses utama yaitu *preprocessing* dan Eksekusi metode. Pada proses Eksekusi metode, maka algoritma *Jaro Winkler* akan diproses untuk melakukan pengkoreksian kata dn memberikan *Suggestion word* yang dimaksud.

3. Hasil dan Pembahasan

3.1 Perancangan *Database*

Perancangan *Database* ini digambarkan dalam bentuk *Entity Relationship Diagram*. [7]

Terdapat beberapa entitas. Perancangan ERD bertujuan untuk memberikan gambaran mengenai *database* yang akan digunakan untuk menyimpan data-data yang diperlukan dalam aplikasi ini seperti ditunjukkan pada Gambar 5.



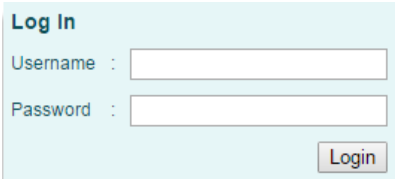
Gambar 5. *Entity Relationship Diagram*

3.2 Tampilan Antarmuka

3.2.1 *Form Login*

Sistem yang dibangun, adalah menggunakan sistem hak akses dengan *login*. Hak Akses *login* dibagi menjadi 2, yaitu *Admin* dan *user*.

Untuk dapat masuk ke sistem, *user* harus melalui *login* dahulu. Hal ini dikarenakan untuk membatasi hak akses. Sehingga *user* yang tidak memiliki hak akses, tidak dapat mengakses sistem tersebut. *Form login* tersebut adalah seperti ditunjukkan pada Gambar 6



Gambar 6 *form login*

3.2.2 Proses Sistem

Sistem yang dibangun adalah aplikasi pengkoreksian dan *Suggestion word* berdasarkan *keyword* yang diinputkan. Pada umumnya, system pencarian judul buku berdasarkan *keyword*, akan menampilkan judul buku sesuai kecocokan kata yang dimasukkan oleh *user*. Namun, sering juga *user* salah menginputkan *keyword*. Misalnya salah mengetikkan kata kunci yang dimaksud, karena *typo*. Hal ini menyebabkan pencarian jurnal ataupun buku tidak diketemukan. Sehingga system akan mengarahkan untuk memberikan koreksi dan *Suggestion word* terhadap *keyword* yang dimaksud.

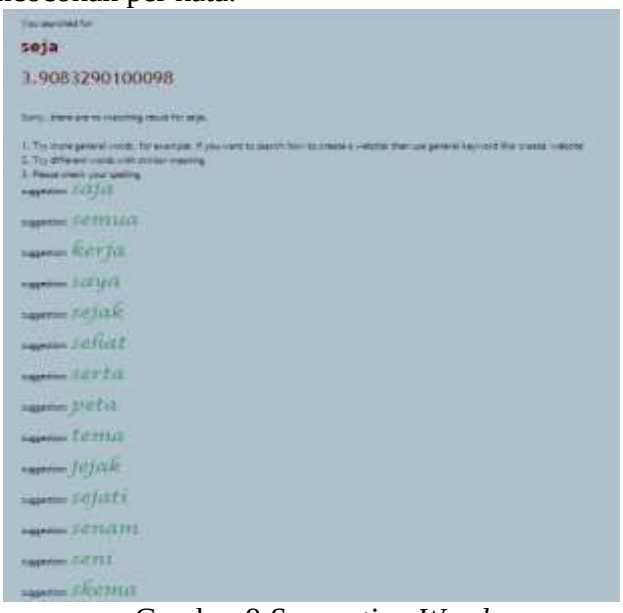
Pada Gambar 7, *user* dapat memasukkan *keyword* untuk mencari judul buku yang dicari. Sehingga akan muncul kata yang diinputkan dan judul buku yang dimasukkan.

Gambar 7 Penginputan *Keyword*

Saat judul buku yang diinputkan tidak sesuai dengan judul buku yang dicari, maka sistem akan menampilkan judul buku sesuai dengan kemiripan kata *keyword* yang diinputkan (judul buku). Atau pada saat *keyword* yang dimasukkan tidak sesuai dengan penulisan, maka aplikasi akan mengarahkan untuk memberikan *Suggestion word* dan pengkoreksian *keyword*.

Sebagai contoh, pada Gambar 8, jika terjadi kesalahan pengetikan ataupun terdapat kata yang mendekati kata kunci yang diketikan, maka hasilnya adalah *Suggestion word* seperti *keyword* yang dimaksud. *Sugestion word* akan ditunjukkan seperti pada Gambar 8.

Sebagai contoh *keyword* yang dimasukkan adalah kata "seja". Seperti yang ditunjukkan pada Gambar Sebenarnya, *keyword* yang dimaksud adalah kata "sejak", Namun pada *database* tidak menyimpan kata tersebut. Sehingga, akan memberikan beberapa kata yang memiliki kemiripan kata (*Suggestion word*). Cara kerja dari *Jaro Winkler* adalah pencocokan kata per tiap huruf. Sehingga jika *keyword* yang dimasukkan tidak cocok, maka akan memberikan beberapa kata pilihan sesuai pencocokan per kata.



Gambar 8 *Suggestion Word*

Seperti yang ditunjukkan pada Gambar 9, maka *keyword* "seja", akan memberikan *Suggestion word* beberapa, yaitu kata "*saja, semua, kerja, saya, sejak, sehat, serta, peta, tema, sejati, senam, seni, skema, self, sama,serap"* Sehingga dari beberapa kata yang dimaksud, memberikan *output* pengkoreksian kalimat dan memberikan *Suggestion word* karena kesalahan pengetikan. Semua *Suggestion word* yang ditampilkan, dapat diklik, sehingga dapat ditunjukkan beberapa judul buku yang dicari sesuai *Suggestion word* yang diberikan.

Gambar 9 *Link* Salah Satu *Suggestion word* dan Pengkoreksian Kata "seja"

Jaro Winkler memiliki cara kerja kemiripan kata sesuai urutan index array yang terdapat pada *keyword* yang dimasukkan. Sebagai contoh, jika kata yang dimasukkan adalah kata "letak", maka *Suggestion word* yang muncul adalah kata "sejak". Karena kata "sejak", huruf "e" dan "a", sesuai dengan array ke 1 dan ke 3 pada kata "letak". Sehingga kata sejak, merupakan salah satu *Suggestion word*.

3.3 *Pseudocode Jaro Winkler*

```
function getCommonCharacters(str1,str2:string)

DEKLARASI
a,b,I,j : integer
distance : integer
temp_str : string
commonchr : string
nomatch : boolean

ALGORITMA
a <- strlen(str1)
b <- strlen(str2)
temp_str <- str2
distance <- floor(min(a,b)/2.0)
{jarak yang dianggap matching dari teori Winkler}

{mencari karakter yang sama dari kedua buah string}
for I <- 0 to a do
  nomatch <- true
  for j <- max (0,i-distance) to nomatch and
```

```
  min (i+distance+1,b)
    if temp_str[j] = str1[i]
      nomatch <- false
      commonchr <- commonchr,str1[i]
      temp_str <- delete(temp_str[j])
    endif
  endfor
  endfor
endfunction
```

Gambar 10 Pseudocode *Jaro Winkler*

Gambar 10 adalah *pseudocode function getCommonCharacters* yang merupakan langkah pertama dari algoritma *Jaro-Winkler. Function* ini digunakan untuk mencari karakter yang sama antara *keyword* dengan data buku, kemudian melakukan pengkoreksian jika ada kata yang tidak tepat dalam pengetikan, dan memberikan s*uggestion word* untuk pencarian judul bahan pustaka yang lain, melalui sebuah link.

4 Kesimpulan dan Saran

4.1 Kesimpulan

Berdasarkan dari pembahasan yang telah dipaparkan, maka dapat disimpulkan bahwa

1) Algoritma *Jaro Winkler* dapat diimplementasikan untuk pengkoreksian pada sebuah *keyword* bahan pustaka.
2) Algoritma *Jaro Winkler* dapat diterapkan untuk memberikan *suggestion word*, jika ada salah pengetikan atau *typo* sata pencarian bahan pustaka
3) *Suggestion word* yang diberikan dapat langsung menghuungkan dengan judul buku yang dimaksud

4.2 Saran

Saran-saran yang dapat dilakukan untuk pengembangan aplikasi ini adalah sebagai berikut :

1) Penggabungan beberapa algoritma dengan algoritma Jaro-Winkler untuk pencarian yang lebih akurat dengan menerapkan SEO.
2) Menggunakan sistem *Text Mining* yang kompleks untuk hasil yang lebih akurat

5 Daftar Pustaka

[1]. Kurniawati, Anna, "Implementasi Algoritma Jaro-Winkler Distance Untuk Membandingkan Kesamaan Dokumen Berbahasa Indonesia, Proceeding, 2014.

[2]. Kornain, Ahmad & Ferry Yansen., Penerapan Algoritma Jaro-Winkler Distance Untuk Sistem Pendeteksi Plagiarisme Pada Dokumen Teks Berbahasa Indonesia, Palembang., 2014.

[3]. Adriyani, dkk. Implementasi Algoritma Jaro-Winkler Distance untuk Membandingkan Kesamaan Dokumen Berbahasa Indonesia, Palembang. 2012.
[4]. Junedy, Richard, Perancangan Aplikasi Deteksi Kemiripan Isi Dokumen Teks Dengan Menggunakan Metode Levenshtein Distance. Medan, 2014
[5]. Rokhmah, Dewi Pyriana & Suprapto. "Program Aplikasi Editor Kata Bahasa Indonesia Menggunakan Metode Approximate String Matching Dengan Algoritma levenshtein Distance Berbasis Java", Proceeding, 2013
[6]. Pressman, R.S., "Software Enginering (A Practitioner's Approach)", 5th Ed.,Prentice-Hall International, Inc, 2010.
[7]. Connolly, T., & Begg, C. "Database Systems : A Practical Approach to Design, Implementation, and Management (5th ed.). English: Addison Wesley", 2010