IMPLEMENTASI METODE *LEVENSTHEIN DISTANCE* UNTUK PENCARIAN *KEYWORD* PADA BAHAN PUSTAKA

Kristien Margi S[1], **Henny Hartono[2], **Agus Toni[2]

(1) Teknik In*form*atika Universitas Bunda Mulia, (08562681855, kksuryaningrum@bundamulia.ac.id)

(2) Sistem In*form*asi Universitas Bunda Mulia, (08158012772, hhartono@bundamulia.ac.id)

(3) Teknik In*form*atika Universitas Bunda Mulia, (085772836966, cibuy731@gmail.com)

Abstrak

Perpustakaan adalah suatu unit kerja yang berupa tempat menyimpan koleksi bahan pustaka yang diatur secara sistematis dan dapat digunakan oleh pemakainya sebagai sumber in*form*asi. Salah satu fungsi perpustakaan adalah sebagai tempat penyimpanan. Perpustakaan bertugas menyimpan koleksi (in*form*asi) karena tidak mungkin semua koleksi dapat dijangkau oleh perpustakaan.Perpustaakn berisi kumpulan buku-buku koleksi yang dikelola secara rapi dan teratur. proses pencarian pustaka berdasarkan *keyword* judul buku atau nama pengarang. *Keyword* yang diketikkan akan merujuk pada judul buku yang dicari. Sehingga judul buku yang tidak mengandung kata yang diketikkan pada *keyword* tersebut tidak akan muncul dalam proses pencarian. Namun seiring berkembangnya zaman, *user* membutuhkan suatu aplikasi yang membantu dalam proses pencarian bahan pustaka yang tidak hanya berdasarkan judul buku atau pengarang saja. Oleh karena itu dibutuhkan suatu aplikasi yang akan menghasilkan sebuah *output* dengan menampilkan beberapa judul buku yang memiliki kemiripan kata dengan *keyword* yang diketikkan. Penerapan sistemnya dengan menerapkan metode *Levenshtein*. Proses kerja metode *Levenshtein* adalah dengan pemrosasan jumlah minimal operasi yang dibutuhkan untuk mengubah suatu *string* ke *string* yang lain, di mana operasi-operasi tersebut adalah operasi penyisipan, penghapusan, atau penyubstitusian sebuah karakter.

Kata kunci : *Levenshtein, searching, perpustakaan, klasifikasi*

1. Pendahuluan

Pada zaman global sekarang, pendidikan merupakan sesuatu yang penting. Karena pendidikan merupakan akar dari peradaban sebuah bangsa. Pendidikan sekarang telah menjadi kebutuhan pokok yang harus dimiliki setiap orang agar bisa menjawab tantangan kehidupan. Untuk memperoleh pendidikan, banyak cara yang dapat dicapai. Diantaranya melalui perpustakaan. Karena di perpustakaan berbagai sumber in*form*asi bisa diperoleh, selain itu banyak juga manfaat lain yang dapat diperoleh melalui perpustakaan. Ketika mendengar kata perpustakaan, yang dipikirkan adalah langsung terbayang sederetan buku-buku yang tersusun rapi di dalam rak sebuah ruangan. Pendapat ini kelihatannya benar, tetapi kalau diperhatikan lebih lanjut, hal itu belumlah lengkap. Karena setumpuk buku yang diatur di rak sebuah toko buku tidak dapat disebut sebagai sebuah perpustakaan.

Perpustakaan adalah tempat untuk mengembangkan informasi dan pengetahuan yang dikelola oleh suatu lembaga pendidikan, sekaligus sebagai sarana edukatif untuk membantu memperlancar cakrawala pendidik dan peserta didik dalam kegiatan belajar mengajar.

Pada mulanya setiap ada kumpulan buku-buku koleksi yang dikelola secara rapi dan teratur disebut perpustakaan, tetapi karena adanya perkembangan teknologi modern dalam usaha pelestarian dan pengembangan in*form*asi, maka koleksi perpustakaan tidak hanya terbatas buku-buku saja tetapi juga beraneka ragam jenisnya. Perpustakaan biasanya berisi bahan pustaka yang berupa buku, jurnal ataupun skripsi yang tersimpan.

Pada umumnya perpustakaan menyajikan, bahan pustaka, dalam bentuk *hard copy*. Pengunjung hanya bisa melakukan pencarian pustaka yang dibutuhkan sesuai kode yang ditempel pada masing-masing bahan pustaka, atau judul bahan pustka pada masing-masing bahan pustaka. Setelah itu mencari sesuai nomor ID pustaka. Sistem yang digunakan untuk pencarian di perpustakaan umumnya adalah berdasarkan judul pada masing-masing pustaka. Sistem yang digunakan biasanya hanya melakukan pencarian berdasarkan *keyword* yang berupa judul buku. Jika judul pustaka yang dimasukkan tidak terdapat dalam *database* perpustakaan tersebut tidak cocok seperti *keyword* yang dimasukkan pada sistem pencarian pustaka, maka buku atau bahan pustaka yang dicari tidak ditemukan.

Namun, kebutuhan sekarang ini terus meningkat, terkadang *user* membutuhkan suatu sistem yang dapat melakukan pencarian bahan pustaka secara cepat, tetapi tidak hanya yang menyangkut dengan judul bahan pustaka yang dimasukkan. *User* juga membutuhkan suatu sistem yang bisa melakukan pencarian pustaka yang masih berhubungan dengan judul bahan pustaka yang dicari. Sehingga dibutuhkan suatu sistem yang dapat membantu *user*, untuk melakukan pencarian judul buku, yang tidak hanya sesuai kecocokan "*keyword*" yang dimasukkan, tetapi juga berdasarkan hubungan padanan kata yang ada pada judul pustaka yang dicari melalui kemiripan *keyword*. Dengan adanya sistem ini, diharpakan ketika *user* melakukan pencarian buku yang dicari berdasarkan *keyword* yang dimasukkan maka *user* tidak hanya mendapatkan judul buku yang dicari, tetapi *keyword* yang diproses juga akan menghasilkan *output* semua judul buku yang berhubungan dengan *keyword* tersebut.

Oleh karena itu, dibutuhkan sebuah aplikasi yang dapat menghasilkan sebuah *output* dengan menerapkan algoritma ke dalam pencarian suatu pustaka yang diaplikasikan pada suatu *keyword* sesuai kemiripan *keyword* yang diinputkan. *Output* yang didapat dari sistem ini, tidak hanya berupa judul pustaka seperti *keyword* yang dimasukkan, tetapi juga berdasarkan kemiripan *keyword* yang diinputkan dengan *database* yang tersimpan pada perpustakaan tersebut. Misalnya sesuai kemiripan judul buku dengan abstrak pada pustaka tersebut.

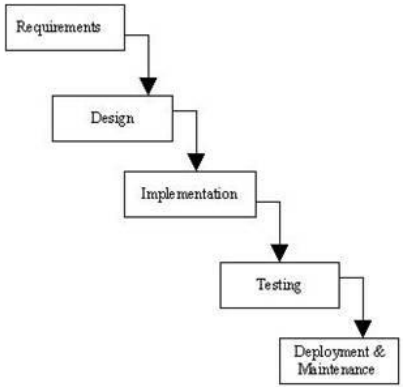
2. Metodologi

2.1 Metodologi Pengembangan Sistem

Aplikasi yang dibangun yaitu merupakan aplikasi yang menghasilkan sebuah aplikasi/sistem yang dapat membantu meningkatkan efisiensi dan memaksimalkan hasil pada algoritma *Levenshtein* untuk mendapatkan sebuah *output* kemiripan kata atau *keyword* yang di*input*kan. Untuk metode pengembangan yang dipilih disesuaikan dengan kebutuhan, yaitu metode pengembangan aplikasi/sistem *waterfall.* [1]

Kelebihan dari metode *waterfall* adalah mencerminkan kepraktisan engineering, proses tidak linear dan sederhana tapi mengandung urutan iterasi dan aktivitas pengembangan. Ketika perangkat lunak telah digunakan terdapat kesalahan dan kelalaian dalam analisa kebutuhan tersebut dapat diatasi. Sedangkan kekurangan dari metode ini adalah, perubahan sulit dilakukan karena sifatnya yang kaku. Karena sifat kakunya ini, model ini cocok ketika kebutuhan dikumpulkan secara lengkap sehingga perubahan bias ditekan sekecil mungkin. Tetapi pada kenyataannya jarang sekali konsumen atau pengguna yang bisa memberikan kebutuhan secara lengkap, perubahan kebutuhan adalah sesuatu yang wajar terjadi. *Database* ini dibuat untuk keperluan sistem *database* yang cepat, handal dan mudah digunakan [1]

Secara detail, alur model *waterfall* yang merupakan model klasik akan digambarkan pada Gambar 1.



Gambar 1 Alur Model *Waterfall* [1]

1. *Requirements/ Analisys*
   Requirements adalah tahapan untuk menentukan kebutuhan data yang akan digunakan untuk membangun sistem, tahapan ini bertujuan untuk menentukan apa yang harus dilakukan untuk mengatasi masalah yang dihadapi. Hal yang terutama dilakukan pada tahap ini adalah menyusun elemen-elemen logikal yang menggambarkan keseluruhan sistem seperti proses dan data. Data-data kemudian secara lengkap dianalisa kelayakannya untuk dijadikan metode dalam pengembangan sistem informasi beserta kebutuhan *database* yang harus dipenuhi oleh program yang akan dibuat. Fase ini dikerjakan agar menghasilkan desain sistem yang lengkap. Data yang dibutuhkan untuk membangun sistem ini adalah id_buku, judul_buku, id_penerbit, nama_penerbit, alamat_penerbit, id_pengarang, nama_pengarang, tahun_terbit, *keyword*, abstrak.
2. *Design*
   Tahapan *design menentukan* bagaimana cara menyelesaikan masalah dengan mengedepankan fokus pada rancangan fisikal seperti struktur data, tampilan layar, *database*. Desain *software* memiliki berbagai tahapan yang berfokus pada atribut program yang jelas yaitu : data *structure*, *software architecture*, *interface representations*, dan detail procedur *(algorithm*). Proses desain menterjemahkan kebutuhan pengguna dalam sebuah dokumen aplikasi yang dapat diperkirakan kualitasnya sebelum proses *coding* dimulai. Pada fase ini dilakukan pembuatan flowchart (alur), arsitektur sistem, DFD dan ERD.
3. *Implementation / Code*
   Tahap implementasi atau coding adalah tahap dimana hasil desain *software* diterjemahkan ke dalam bahasa yang dapat dimengerti oleh komputer. Sehingga pada tahap ini perancangan akan dibangun ke dalam *pseudocode* dan coding-coding melalui PHP dan My SQL.

4. Testing

Tahapan testing dilakukan setelah dibuat program aplikasi untuk mengetahui apakah sudah dapat berfungsi seperti yang diinginkan, kekurangan apa yang belum dan sistem apa saja yang harus dibangun. Setelah melewati tahapan pengujian sistem akhirnya diimplementasikan kepada *user*. Sistem pengujian menggunakan *white box* dan *black box*.

5. Maintenance

Tujuan dari tahapan ini adalah melakukan dukungan dan pemeliharaan terhadap implementasi sistem agar dapat tetap berfungsi pada tingkat yang lebih tinggi. Sehingga dapat dilakukan perbaikan jika ada yang tidak sesuai.

2.2. Metode *Levensthein*

Menurut Smith (2015), Metode *Levenshtein Distance* dapat digunakan untuk melakukan perhitungan beda jarak antara dua *string*. Jarak atau *distance* adalah jumlah minimum dari operasi hapus, *insert* atau substitusi yang dibutuhkan untuk merubah *string* asal (s) menjadi *string* target (t) [2][3]

Sebagai contoh:

- Jika s adalah "coba" dan t adalah "coba", maka LD(s,t) = 0, dikarenakan tidak ada trans*form*asi yang dibutuhkan. Kedua *string* adalah identic
- Jika s adalah "coba" dan t adalah "coka", maka LD(s,t) = 1, dikarenakan satu substitusi (merubah "b" ke "k") dicukupkan untuk mentrans*form* s menjadi t.

Perhitungan harga pengeditan pada setiap operasi yang dilakukan adalah sesuai
dengan aturan berikut

- d(a,ε) = 1 harga untuk menghapus sub*string* a
- d(ε,a) = 1 harga untuk penyisipan sub*string* a
- d(a,b) = 1 harga untuk substitusi sub*string* a ke sub*string* b
- d(a,a) = 0

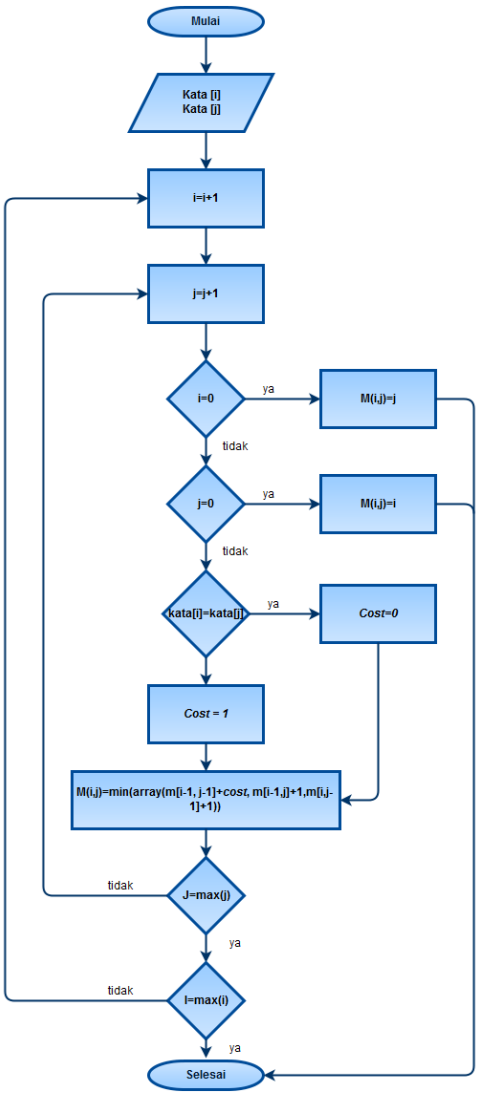
Semakin besar angka yang dihasilkan oleh operasi *Levenshtein Distance* maka semakin besar perbedaan di antara kedua *string* tersebut [2][4]

Algoritma *Levenshtein Distance*

1. m = panjang kata1 (kata kunci yang di*input*kan oleh *user*).
2. n = panjang kata2 (kata dari tabel kata yang digunakan untuk pembanding).
3. d[0,0] = 0.
4. Untuk i = 1 sampai m, kerjakan d[i,0] = d[i-1,0] + c(kata1i, ε).
5. Untuk j = 1 sampai n, kerjakan d[0,j] := d[0,j-1] + c(ε, kata2j).
6. Menghitung nilai *array* dalam matrik
   Untuk i = 1 sampai m, kerjakan,
   Untuk j = 1 sampai n, kerjakan
   d[i,j] = d[i-1,j] + c(kata1i, ε)
   Min d[i,j-1] + c(ε, kata2j)
   d[i-1,j-1] + c(kata1i, kata2j)

Hasil perhitungan jarak ditunjukkan pada kolom d[n,m].

Gambar 2 adalah *flowchart* untuk pencarian *distance* antara *keyword* dengan data buku pada *database*, dan merupakan cara kerja algoritma *Levenshtein Distance*.



Gambar 2 *Flowchart Levenshtein Distance* [4]

Langkah-langkahnya sebagai berikut:

- Pembuatan metrik dengan panjang baris = panjang kata[i], dan lebar = panjang kata[j].
- Lalu isi *cell* [i+1,j+1], dengan ketentuan:

1. Jika i = 0, maka M(i,j) = j, jika tidak periksa j.
2. Jika j = 0, maka M(i,j) = i, jika tidak maka periksa apakah kata[i] = kata[j].
3. Jika kata[i] = kata[j] maka *cost* = 0, jika tidak maka *cost* = 1.
4. Lalu cari nilai yang paling kecil/ *minimum* antara *cell* bagian pojok kiri atas(*cell* [i-1, j-1] + *cost*), *cell*

bagian kiri (*cell*[i-1, j]+1) atau *cell* bagian bawah (*cell*[I, j-1]+1).

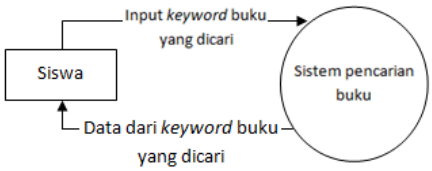
5. Periksa apakah J=*max*(j), jika iya periksa apakah I=*max*(i). Jika J != *max*(j) maka J = j+1, lalu ulangi langkah ke 2.

Jika I = *max*(i) maka selesai, jika I != *max*(i) maka I = i+1 lalu ulangi langkah ke 1.

2.3. *Data Flow Diagram*

DFD adalah alat pembuatan model yang memberikan penekanan hanya pada fungsi sistem. DFD ini merupakan alat perancangan sistem yang berorientasi pada alur data dengan konsep dekomposisi dapat digunakan untuk penggambaran analisa maupun rancangan sistem yang mudah dikomunikasikan oleh profesional sistem kepada pemakai maupun pembuat program.[1] [5]

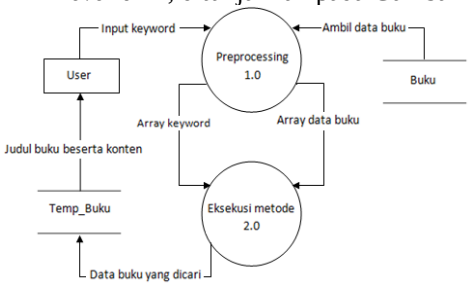
Diagram context aplikasi ini dapat digambarkan pada gambar 3.



Gambar 3 Diagram Context

Berdasarkan pada Gambar 3 terdapat satu entitas yang akan berinteraksi dengan sistem yang akan dibangun, yaitu entitas *user*. *User* berperan sebagai entitas luar yang akan berinteraksi langsung dengan sistem.

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

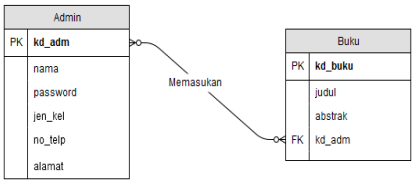


Gambar 4 DFD *Level* 0

Pada diagram *level* 0 yang terbentuk, terdapat 2 proses utama yaitu *preprocessing* dan Eksekusi metode.

2.4 *Entity Relationship Diagram*

Hubungan antar entitas atau ERD pada perancangan aplikasi ini, ditunjukkan pada Gambar 5

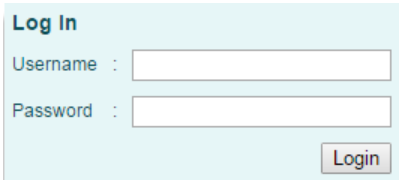


Gambar 5 *Entity Relationship Diagram* Perancangan Sistem

3. Pembahasan

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



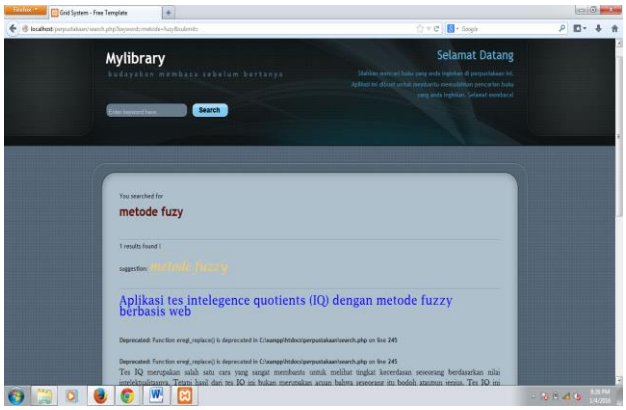
Gambar 6 *Form Login*

3.1 Proses Sistem

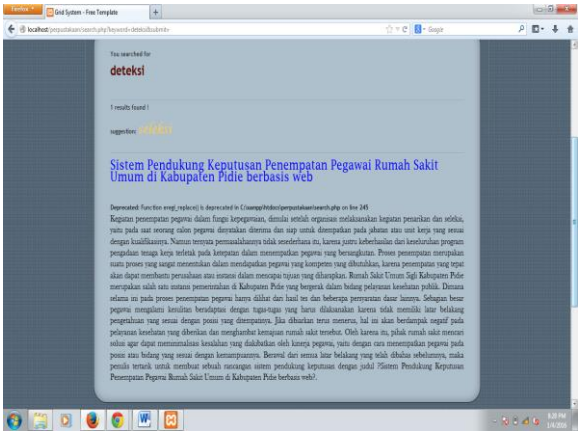
Setelah masuk ke *form login*, maka *user* dapat melakukan proses pencarian buku atau bahan pustaka Sistem yang dibangun adalah aplikasi pencarian buku berdasarkan *keyword* yang diinputkan. Pada umumnya, aplikasi pencarian judul buku sesuai *keyword*, akan menampilkan judul buku sesuai kecocokan kata yang dimasukkan oleh *user* pada labael "*keyword*". Namun, pada aplikasi ini akan menampilkan judul buku, sesuai dengan pencocokan kata atau kemiripan kata pada *keyword* judul buku. Sehingga akan ditampilkan sesuai kemiripan kata yang ada pada judul buku dan abstrak yang disimpan pada *database*.

Proses pencarian judul buku yang dibutuhkan, akan diproses berdasarkan kemiripan judul buku dan abstrak yang telah disimpan oleh *database*.Cara kerja metode *levensthein Distance* memproses, jika kata kunci yang diketikkan benar dan juga tidak ada kata pada *database* yang mendekati dengan kata kunci yang diketikan maka akan menghasilkan ouput beberapa judul buku.

Pada Gambar 7, kata kunci yang diketikan adalah "metode *fuzy*", berarti di dalam *database* buku, tidak ada buku lain yang terkait dengan kata kunci "metode *fuzy*" selain buku dengan judul "Aplikasi Tes *Intelegence Quotients* (IQ) dengan Metode *Fuzy* Berbasis *Web*". Jika hasil pencarian yang ditemukan lebih dari 3 data buku, maka hasil pencarian akan ditampilkan per halaman 3 buku, dengan halaman sebanyak 3 dibagi dengan jumlah data yang ditemukan.



Gambar 7 *Form* Proses Sistem



Gambar 8 Penyeleksian Judul Buku sesuai Kemiripan Kata

Sistem akan melakukan pencarian berdasarkan judul buku yang tersimpan pada *database*. *Keyword* yang tersedia adalah kata yang akan digunakan untuk pencarian judul buku atau judul pustaka, dan kemiripan judul pustaka yang dicari.

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

Sebagai contoh judul buku yang diinputkan adalah kata "deteksi", walaupun di perpustakaan tersebut tidak terdapat judul buku yang mengandung kata deteksi, maka program akan tetap menampilkan beberapa judul buku yang memiliki kemiripan kata seperti kata "deteksi". Hal ini dapat dilihat berdasarkan kemiripan kata seperti yang ditunjukkan pada abstrak pada buku yang dimaksud. Kemiripan kata yang dicocokkan dengan *keyword* walaupun tidak sesuai dengan judul buku, maka akan dicocokkan sesuai dengan kemiripan kata pada abstrak yang sudah tersimpan di *database* sistem tersebut. Proses pencarian buku ini dapat dilihat pada Gambar 8.

Pada gambar 8, ditunjukkan *output* pencarian buku walaupun tidak sesuai dengan judul buku yang dimaksud. Sehingga dari sistem tersebut akan menghasilkan *output* judul buku yang abstraknya sesuai dengan *keyword*. *Keyword* yang diinputkan oleh *user* akan di-*highlight* dengan warna kuning supaya mempermudah *user* untuk menemukan *keyword* yang dimaksud.

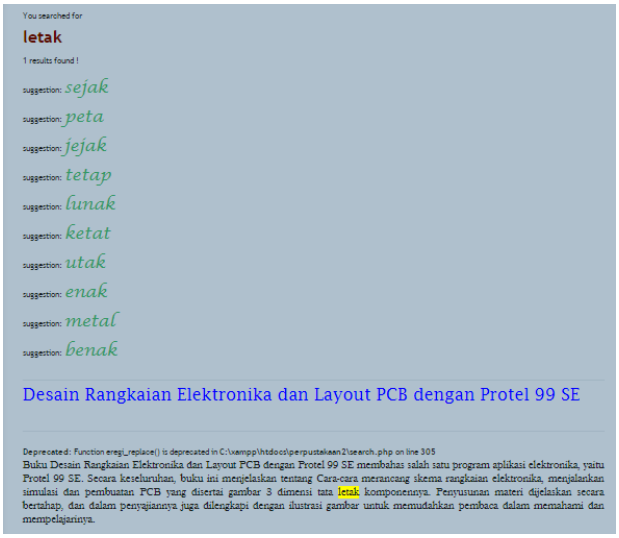
Pada sistem aplikasi in, juga dilengkapi sistem yang memberikan *suggestion word*. Jika terjadi kesalahan pengetikan ataupun terdapat kata yang mendekati kata kunci yang diketikan, maka hasilnya adalah *suggestion word* seperti *keyword* yang dimaksud. *Sugestion word* akan ditunjukkan seperti pada Gambar 9.

Keyword yang dimasukkan adalah kata "letak". Namun pada *database* tidak menyimpan kata tersebut. Sehingga, akan memberikan beberapa kata yang memiliki kemiripan kata (*suggestion word*). Cara kerja dari *levensthein distance* adalah pencocokan kata per tiap huruf. Sehingga jika *keyword* yang dimasukkan tidak cocok, maka akan memberikan beberapa kata pilihan sesuai pencocokan per kata.

Seperti yang ditunjukkan pada Gambar 9, maka *keyword* "letak", akan memberikan *suggestion word* beberapa, yaitu kata sejak, peta, jejak, tetap, lunak, ketat, utak, enak, metal, benak. Kata yang memiliki kemiripan adalah yang di bold. Sehingga dari beberapa kata yang dimaksud, memberikan *output* judul buku "Desain Rangkaian Elektronika dan Layout PCB dengan Protel 99 SE", sesuai dengan abstrak yang memiliki kata "letak".

Lavensthein distance 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*.

Suggestion word yang ditampilkan adalah sebuah *link*, jika pengguna aplikasi ingin mencari buku sesuai dengan salah satu *suggestion word* yang diberikan.



Gambar 9 Suggestion Word

4. Kesimpulan

Beberapa kesimpulan yang dihasilkan dari penelitian ini adalah

1. Metode *Levensthein* dapat diterapkan untuk menampilkan judul buku yang dicari walaupun bukan *keyword* yang diinputkan. Hal ini dibuktikan berdasarkan kemiripan kata sesuai kata dasar yang diinputkan dan memberikan *suggestion word* berdasarkan kemiripan kata
2. Jumah kata yang diinputkan dan dicocokkan bisa dicocokkan dengan minimal 1 kata, dan maximal 3 kata.

Daftar Pustaka

[1] Pressman, R.S., "*Software Enginering (A Practitioner's Approach)"*, 5th Ed.,Prentice-Hall International, Inc, 2001.

[2] Rokhmah, Dewi Pyriana & Suprapto. "*Program Aplikasi Editor Kata Bahasa Indonesia Menggunakan Metode Approximate String Matching Dengan Algoritma levenshtein Distance Berbasis* Java*"*, Proceeding, 2013

[3] Junedy, Richard, *Perancangan Aplikasi Deteksi Kemiripan Isi Dokumen Teks Dengan Menggunakan Metode Levenshtein Distance*. Medan, 2014

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

[5] Connolly, T., & Begg, C. "*Database* Systems : A Practical Approach to Design, Implementation, and Management (5th ed.). English: Addison Wesley", 2010