Implementasi Algoritma Levenshtein Pada Sistem Pencarian Judul Skripsi/Tugas Akhir

Ida Bagus Ketut Surya Arnawa
STIKOM Bali
Jl.Raya Puputan No. 86 Renon, Denpasar-Bali
e-mail: arnawa@stikom-bali.ac.id

Abstrak

Dalam penyusunan laporan Skripsi / Tugas Akhir mahasiswa memerlukan banyak referensi baik berupa buku maupun laporan Skripsi / Tugas Akhir. Mahasiswa dapat mencari referensi tersebut di perpustakaan STIKOM Bali dengan menggunakan sistem pencarian. Seringkali, terjadi kesalahan dalam menginputkan kata kunci yang bukan merupakan ejaan yang baku atau salah ketik. Sebagai contoh kata kunci "resiko", padahal ejaan bakunya adalah "risiko". Tentu saja mahasiswa akan memperoleh informasi yang kurang lengkap dan bahkan mahasiswa gagal dalam mendapatkan informasi yang sesuai dengan kata kunci yang di masukan. Untuk mengatasi permasalahan yang dialami mahasiswa dalam melakukan pencarian judul Skripsi / Tugas Akhir, maka diperlukan suatu metode pendekatan pencarian string agar hasil pencarian dapat maksimal. Salah satu algoritma yang dapat digunakan adalah Levenshtein yang dapat menghitung jarak keterbedaan antara dua string. Implementasi algoritma levenshtein pada sistem pencarian judul Skripsi/Tugas Akhir sudah dapat mengatasi permasalahan pada kesalahan ejaan kata kunci dengan mekanisme penambahan, penyisipan dan penghapusan karakter.

Kata kunci: *Pencarian, Levenshtein, Skripsi, Tugas Akhir*

Abstract

In preparing the report Thesis / Final student requires a lot of references either in the form of books and reports Thesis / Final. Students can search the library reference STIKOM Bali by using the search system. Often, there was an error in the input keyword is not a standardized spelling or typos. For example, the keyword "resiko", while spelling default is "risiko". Of course, students will get the missing information and even a student fails in getting the information according to keywords in the input. To overcome the problems experienced by students in conducting title searches Thesis / Final, we need a method string search approach that search results can be maximized. One algorithm that can be used is to calculate Levenshtein distance between two strings. Levenshtein algorithm implementation on the title search system Thesis / Final've been able to solve problems with spelling mistakes keywords with the mechanism of the addition, insertion and deletion of characters.

Keywords*: Search, Levenshtein,Thesis, Final Project*

1. Pendahuluan

STIKOM Bali merupakan salah satu Sekolah Tinggi Manajemen Informatika dan Teknik Komputer yang berada di pulau Bali. STIKOM Bali diresmikan pada tanggal 10 Agustus 2002 dengan ijin mendiknas RI No. 157/D/O/2002. Sebagai salah satu Sekolah Tinggi Manajemen Informatika dan Teknik Komputer STIKOM Bali memiliki jumlah mahasiswa yang cukup banyak dan hampir setiap tahun mengalami peningkatan. Secara otomatis, jumlah mahasiswa yang semakin banyak tersebut akan meningkatkan jumlah lulusan dan jumlah laporan Skripsi / Tugas Akhir pada STIKOM Bali.

Dalam penyusunan laporan Skripsi / Tugas Akhir mahasiswa memerlukan banyak referensi baik berupa buku maupun laporan Skripsi / Tugas Akhir dari penelitian yang sebelumnya. Untuk referensi laporan Skripsi / Tugas Akhir mahasiswa dapat mencari di perpustakaan STIKOM Bali. Dalam melakukan pencarian mahasiswa dapat menggunakan sistem yang sudah tersedia, dengan cara memasukkan kata kunci yang ingin di cari maka secara otomatis sistem akan menampilkan judul Skripsi / Tugas Akhir yang sesuai dengan kata kunci yang dimasukan. Seringkali, dalam melakukan pencarian mahasiswa menginputkan kata kunci yang bukan merupakan ejaan yang baku atau salah ketik. Sebagai contoh mahasiswa mengetikkan kata kunci "resiko", padahal ejaan bakunya adalah "risiko". Tentu saja

mahasiswa akan memperoleh informasi yang kurang lengkap dan bahkan mahasiswa gagal dalam mendapatkan informasi yang sesuai dengan kata kunci yang di masukan.

Untuk mengatasi permasalahan yang dialami mahasiswa dalam melakukan pencarian judul Skripsi / Tugas Akhir, maka diperlukan suatu metode pendekatan pencarian *string* agar hasil pencarian dapat maksimal. Ada beberapa algoritma yang dapat diimplementasikan dalam memberikan kata saran yang paling mendekati dari kata yang salah pengetikannya. Salah satu algoritma yang dapat digunakan adalah *Levenshtein* yang dapat menghitung jarak keterbedaan antara dua *string* [2] . Algoritma ini digunakan secara luas dalam berbagai bidang, misalnya pengecekan ejaan, mesin pencarian, analisis DNA dan lain-lain. Berdasarkan hal yang telah diuraikan di atas maka, penulis berniat untuk melakukan penelitian untuk implementasi algoritma *Levenshtein* dalam sistem pencarian judul Skripsi / Tugas Akhir.

2. Tinjauan Pustaka

2.1 Algoritma Levenshtein

Algoritma *Levenshtein* ditemukan oleh ilmuan asal Rusia bernama Vladimir Levenshtein pada tahun 1963, algoritma ini juga disebut dengan algoritma *Edit Distance*. Perhitungan *edit distance* didapatkan dari matriks yang digunakan untuk menghitung jumlah perbedaan *string* antara dua *string*, sebagai contoh hasil penggunaan algoritma ini, *string* "komputer" dan "*computer*" memiliki *distance* 1 karena hanya perlu dilakukan satu operasi saja untuk mengubah satu *string* ke *string* yang lain. Dalam kasus dua *string* di atas, *string* "*computer*" dapat menjadi "komputer" hanya dengan melakukan satu penukaran karakter "c" menjadi "k" [2].

Algoritma *Levenshtein* digunakan secara luas dalam berbagai bidang, misalnya mesin pencari, pengecek ejaan (*spell checking*), pengenal pembicaraan (*speech recognition*), pengucapan dialek, analisis DNA, pendeteksi pemalsuan, dan lain-lain. Algoritma ini menghitung jumlah operasi *string* paling sedikit yang diperlukan untuk mentransformasikan suatu *string* menjadi *string* yang lain [1]. Algoritma *Levenshtein* bekerja dengan menghitung jumlah minimum pentranformasian suatu *string* menjadi *string* lain yang meliputi penghapusan, penyisipan, dan penukaran.

Selisih perbedaan antar *string* dapat diperoleh dengan memeriksa apakah suatu *string* sumber sesuai dengan *string* target. Nilai selisih perbedaan ini disebut juga *edit distance* atau jarak *Levenhstein*. Jarak *Levenshtein* antar *string* "s" dan *string* "t" tersebut adalah fungsi D yang memetakan (s,t) ke suatu bilangan *real* non negatif, sebagai contoh diberikan dua buah *string* s = s(1)s,(2),s(3),...,s(m) dan t = t(1),t(2),t(3),...,t(n) dengan | s | = m dan | t | = n sepanjang alfabet V berukuran r sehingga "s" dan "t" anggota dari V*. s(j) adalah karakter pada posisi ke-j pada *string* "s" dan t(i) adalah karakter pada posisi ke-i pada string "t". Sehingga jarak Levenshtein dapat didefinisikan sebagai (Harahap, 2013).

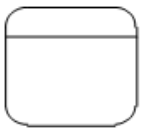
$$D(s,t) = d(s1,t1) + d(s2,t2) + \ldots + d(s1,t1)$$

D(s,t) adalah banyaknya operasi minimum dari operasi penghapusan, penyisipan dan penukaran untuk menyamakan string s dan t. Pada implementasi pencocokan antar string, ketiga operasi tersebut dapat dilakukan sekaligus untuk menyamakan string sumber dengan string target.

2.2 Data Flow Diagram (DFD)

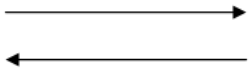
Data Flow Diagram (DFD) memberikan gambaran bagaimana data masuk dan keluar dari dalam dan ke suatu *entity* / representasi dari sumber dan tujuan aliran data tersebut, aturan dari proses data, penyimpanan data, dan entitas eksternal. Selain itu DFD merupakan diagram yang menggambarkan sistem secara terstruktur dengan memecah–mecah menjadi beberapa level dan proses paralel pada sistem serta menunjukkan arus data, simpanan data, kesatuan lain yang ada pada sistem [6][7]. Simbol–simbol yang digunakan dalam menggambarkan *Data Flow Diagram* (DFD) menurut Gane & Sarson antara lain sebagai berikut :

Simbol proses merupakan simbol yang dilakukan oleh orang atau mesin dimana arus data yang ke dalam proses nantinya dihasilkan arus data yang akan keluar.



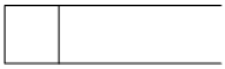
Gambar 1. Simbol Proses

Simbol arus data yang menunjukkan arus data dari data berupa masukan ataupun keluaran.



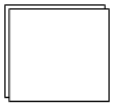
Gambar 2. Simbol Arus Data

Simbol simpanan data merupakan simpanan data dari data dapat berupa *file* atau *database* di sistem komputer. Dari simpanan data di DFD dapat disimbulkan dengan simpanan garis *horizontal pararel* yang tertutup di salah satu ujungnya.



Gambar 3. Simbol Simpanan

Simbol *entity* dari data dapat berupa organisasi atau sistem lainnya berada di lingkungan luarnya akan memberikan *input* atau menerima *output* dari sistem.



Gambar 4. Simbol *Entity*

3. Metode Penelitian

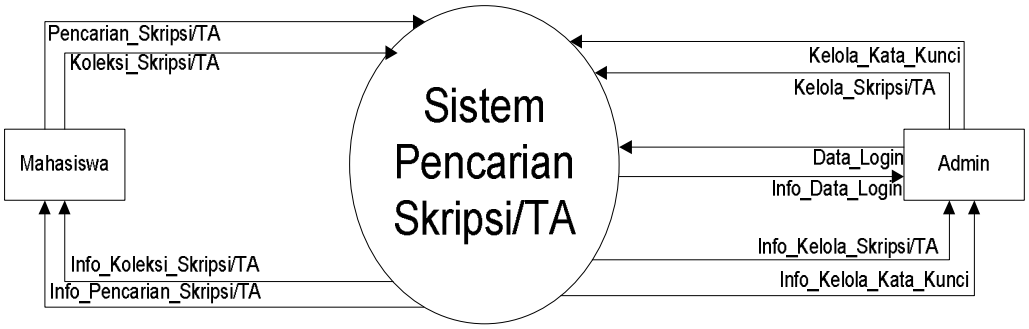
3.1. Tempat dan Waktu Penelitian

Penelitian dilakukan selama 4 bulan di STMIK-STIKOM Bali Jalan Raya Puputan No 86 Renon Denpasar Bali.

3.2. Desain Data Flow Diagram (DFD)

Dengan adanya DFD sistem ini maka proses aliran data dalam sistem dapat diketahui dengan jelas. DFD sistem pencarian judul Skripsi/Tugas Akhir ini dibagi menjadi 3 bagian yaitu DFD Konteks, DFD level 0, dan DFD level 1

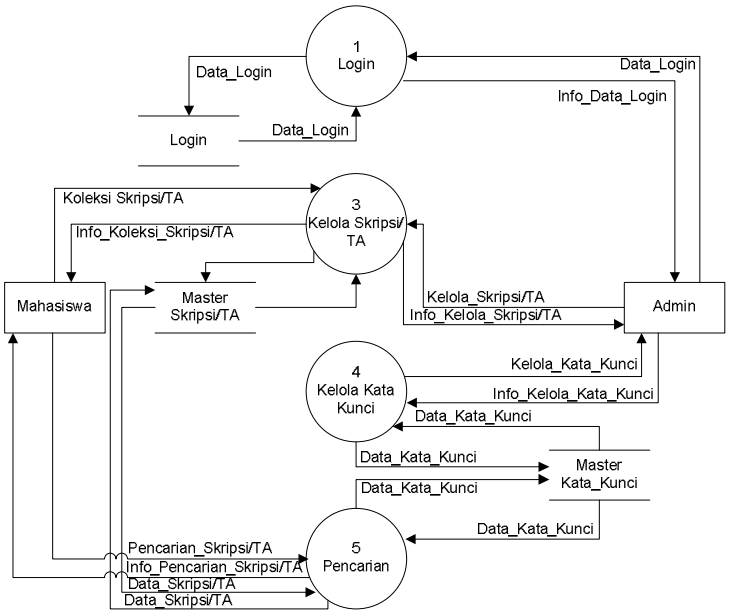
3.2.1 DFD Diagram Konteks



Gambar 5. Diagram Konteks

Gambar 5 menunjukkan diagram konteks yang merupakan gambaran awal *procedural* proses alur data secara keseluruhan. Disini terdapat sebuah sistem yaitu sistem pencarian judul Skripsi/Tugas Akhir. Terdapat 2 entitas luar (*External entity*) yaitu admin dan mahasiswa. Admin melakukan *login* admin dan memasukan *password* terlebih dahulu, lalu admin melakukan kelola data Skripsi/Tugas Akhir dan data kata kunci. Mahasiswa dapat melakukan pencarian data Skripsi/Tugas Akhir dengan cara menginputkan kata kunci yang ingin dicari kemudian sistem akan melakukan pencocokan kata kunci yang tersimpan dengan kata kunci yang diinputkan dan menampilkan informasi sesuai dengan kata kunci.

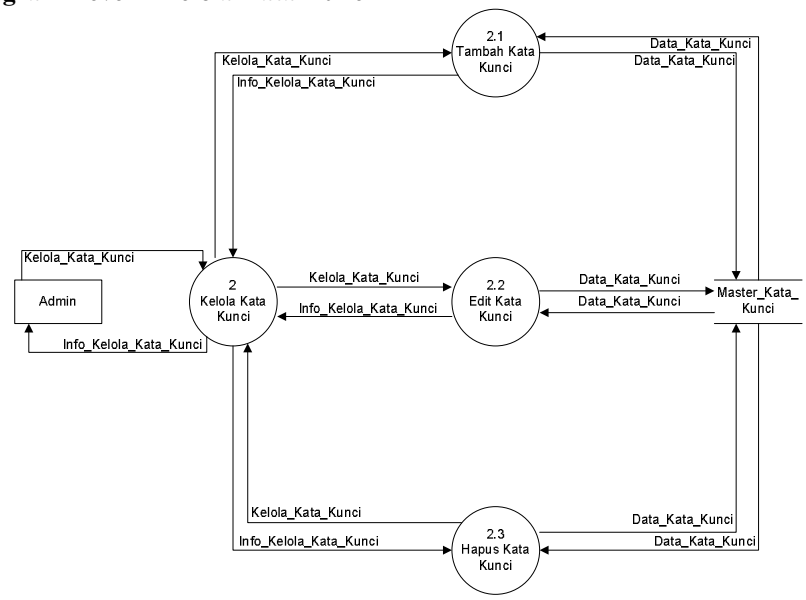
3.2.2 DFD Diagram Level 0



Gambar 6. Diagram Level 0

Gambar 6 menunjukkan proses yang terdapat pada diagram konteks. Proses dipecah menjadi 3, yaitu proses *login*, kelola data dan proses pencarian data. Pada proses *login*, admin melakukan *login* terlebih dahulu sebelum masuk ke dalam sistem. Pada proses kelola data, admin melakukan kelola data Skripsi/Tugas Akhir dan data kata kunci. Pada proses pencarian dapat dilakukan dengan menginputkan kata kunci kemudian sistem akan mencocokkan dengan kata kunci yang sudah tersimpan menggunakan algoritma *levenshtein* kemudian menampilkan hasil pencarian.

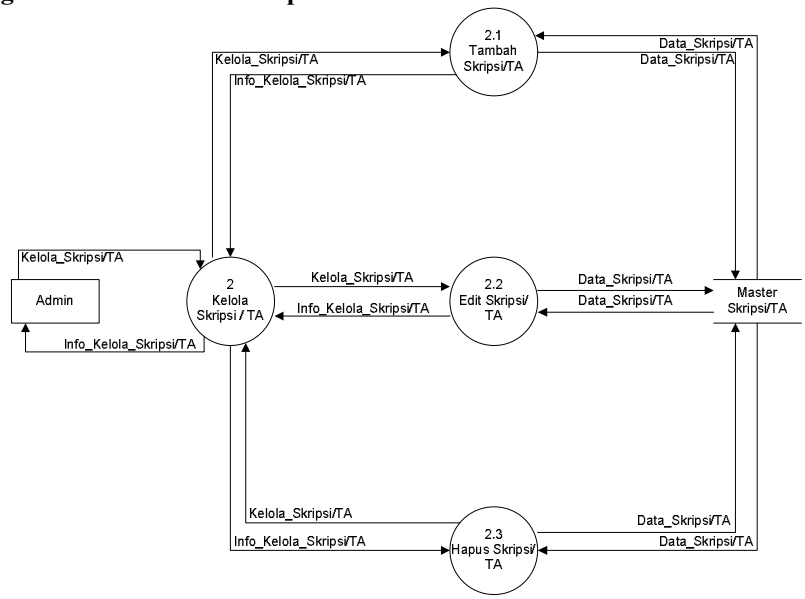
3.2.3 DFD Diagram Level 1 Kelola Kata Kunci



Gambar 7. Diagram Level 1 Kelola Kata Kunci

Gambar 7 menunjukkan pemecahan dari proses kelola data kata kunci. Admin dapat melakukan 3 proses kelola data diantaranya yaitu tambah data kata kunci, edit data kata kunci dan hapus kata kunci.

3.2.4 DFD Diagram Level 1 Kelola Skripsi/TA

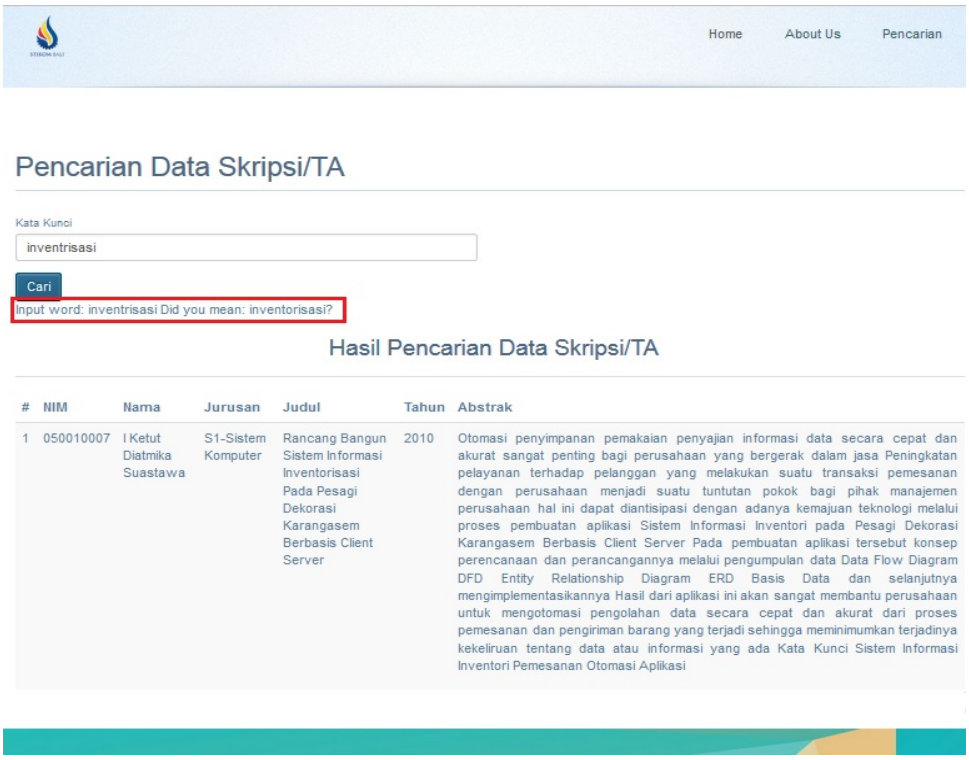


Gambar 8. Diagram Level 1 Kelola Skripsi/TA

Gambar 8 menunjukkan pemecahan dari proses kelola data Skripsi/TA. Admin dapat melakukan 3 proses kelola data diantaranya yaitu tambah data Skripsi/TA, edit data Skripsi/TA dan hapus data Skripsi/TA.

4. Hasil dan Pembahasan

Proses pencarian pada sistem dengan menggunakan kata kunci "inventrisasi", tidak menampilkan hasil dikarenakan kata kunci tersebut tidak sesuai. Pada kondisi seperti ini sistem akan mengganggap kata kunci yang digunakan mengalami kesalahan pada proses pengejaan sehingga sistem memberikan saran kata kunci "inventarisasi" dan hasil pencarian yang menggunakan saran kata kunci dapat dilihat pada Gambar 9.



Gambar.9 Hasil pencarian dengan kata kunci yang tidak sesuai

Pada saat melakukan pencarian pada sistem menggunakan kata kunci yang tidak sesuai yang disebabkan oleh kesalahaan pada pengejaan, maka sistem akan memberikan saran kata kunci yang merupakan hasil dari perbaikan ejaan kata kunci yang sebelumnya. Sistem memberikan saran kata kunci inventorisasi“ yang merupakan perbaikan ejaan dari kata “inventrisasi”. Adapun proses perbaikan ejaan dapat dijelaskan sebagai berikut :

1. Mengkonversi kata kunci “inventrisasi” ke dalam array.
2. Melakukan proses seleksi pada semua kata yang tersimpan pada tabel Kata Kunci. Kata kunci yang digunakan perbandingan adalah kata kunci yang memiliki panjang karakter (P) antara Pkata kunci -3 sampai Pkata kunci + 3. Sehingga kata kunci “inventrisasi” dengan panjang karakter 12, maka kata-kata yang digunakan sebagai pembanding adalah kata yang memiliki panjang karakter diantara 9 – 15.
3. Melakukan perhitungan jarak dengan menggunakan metode Levenshtein terhadap kata kunci “inventrisasi” dengan setiap kata yang terpilih pada tabel Kata Kunci. Sebagai contoh empat kata kunci yang dipilih sebagai perbandingan dan akan dilakukan perhitungan jarak yaitu kata “interpelasi” dengan panjang karakter = 11, kata “inventorisasi” dengan panjang karakter = 13, kata “interupsi” dengan panjang karakter = 9 dan kata “investasi” dengan panjang karakter = 9.

Tabel 1. Menghitung nilai jarak kata kunci dengan kata interpelasi

		I	N	V	E	N	T	R	I	S	A	S	I
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
I	**1**	0	1	2	3	4	5	6	7	8	9	10	11
N	**2**	1	0	1	2	3	4	5	6	7	8	9	10
T	**3**	2	1	1	2	3	4	5	6	7	8	9	10
E	**4**	3	2	2	1	2	3	4	5	6	7	8	9
R	**5**	4	3	3	2	2	3	4	5	6	7	8	9
P	**6**	5	4	4	3	3	3	4	5	6	7	8	9
E	**7**	6	5	5	4	4	4	4	5	6	7	8	9
L	**8**	7	6	6	5	5	5	5	5	6	7	8	9
A	**9**	8	7	7	6	6	6	6	6	6	6	7	8
S	**10**	9	8	8	7	7	7	7	7	7	7	6	7
I	**11**	10	9	9	8	8	8	8	8	8	8	7	6

Tabel 2. Menghitung nilai jarak kata kunci dengan kata inventarisasi

		I	N	V	E	N	T	R	I	S	A	S	I
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
I	**1**	0	1	2	3	4	5	6	7	8	9	10	11
N	**2**	1	0	1	2	3	4	5	6	7	8	9	10
V	**3**	2	1	0	1	2	3	4	5	6	7	8	9
E	**4**	3	2	1	0	1	2	3	4	5	6	7	8
N	**5**	4	3	2	1	0	1	2	3	4	5	6	7
T	**6**	5	4	3	2	1	0	1	2	3	4	5	6
A	**7**	6	5	4	3	2	1	1	2	3	4	5	6
R	**8**	7	6	5	4	3	2	1	2	3	4	5	6
I	**9**	8	7	6	5	4	3	2	1	2	3	4	5
S	**10**	9	8	7	6	5	4	3	2	1	2	3	4
A	**11**	10	9	8	7	6	5	4	3	2	1	2	3
S	**12**	11	10	9	8	7	6	5	4	3	2	1	2
I	**13**	12	11	10	9	8	7	6	5	4	3	2	1

Tabel 3. Menghitung nilai jarak kata kunci dengan kata interupsi

		I	N	V	E	N	T	R	I	S	A	S	I
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
I	**1**	0	1	2	3	4	5	6	7	8	9	10	11
N	**2**	1	0	1	2	3	4	5	6	7	8	9	10
T	**3**	2	1	1	2	3	4	5	6	7	8	9	10
E	**4**	3	2	2	1	2	3	4	5	6	7	8	9
R	**5**	4	3	3	2	2	3	4	5	6	7	8	9
U	**6**	5	4	4	3	3	3	4	5	6	7	8	9
P	**7**	6	5	5	4	4	4	4	5	6	7	8	9
S	**8**	7	6	6	5	5	5	5	5	5	6	7	8
I	**9**	8	7	7	6	6	6	6	6	6	6	7	8

Tabel 4. Menghitung jarak dengan kata investasi

		I	N	V	E	N	T	R	I	S	A	S	I
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
I	**1**	0	1	2	3	4	5	6	7	8	9	10	11
N	**2**	1	0	1	2	3	4	5	6	7	8	9	10
V	**3**	2	1	0	1	2	3	4	5	6	7	8	9
E	**4**	3	2	1	0	1	2	3	4	5	6	7	8
S	**5**	4	3	2	1	1	2	3	4	5	6	7	8
T	**6**	5	4	3	2	2	1	2	3	4	5	6	7
A	**7**	6	5	4	3	3	2	2	3	4	5	6	7
S	**8**	7	6	5	4	4	3	3	3	3	4	5	6
I	**9**	8	7	6	5	5	4	4	4	4	4	5	5

Dari perhitungan yang telah dilakukan melalui Tabel.1, Tabel.2, Tabel.3 dan Tabel.4 maka diperoleh nilai jarak untuk masing-masing kata yang dibandingkan yaitu sebagai berikut :

Nilai jarak (inventrisasi, interpelasi) = 6
Nilai jarak (inventrisasi, inventorisasi) = 1
Nilai jarak (inventrisasi, interupsi) = 8
Nilai jarak (inventrisasi, investasi) = 5

Hasil nilai jarak yang didapatkan kemudian diurutkan mulai dari nilai jarak yang terkecil sampai nilai jarak yang terbesar. Kata yang berada pada urutan teratas atau kata yang memiliki nilai jarak terkecil merupakan kata yang terpilih sebagai kata yang disarankan. Sehingga kata yang terpilih sebagai kata saran pada kasus diatas adalah kata “inventorisasi”.

5. Simpulan

Berdasarkan uraian laporan penelitian diatas yang berjudul “Implementasi Algoritma Levenshtein Pada Sistem Pencarian Judul Skripsi/Tugas Akhir Mahasiswa STMIK STIKOM Bali”, penulis dapat menyimpulkan sebagai berikut:

1. Algoritma Levenshtein dapat membantu mengatasi permasalahan pada kesalahan ejaan kata kunci dengan mekanisme penambahan, penyisipan dan penghapusan karakter.
2. Sistem ini dapat membantu mahasiswa dalam melakukan pencarian judul Skripsi/TA dengan hasil pencarian yang optimal.
3. Pengecekan ini masih sebatas pengecekan kesalahan pada ejaan.

Daftar Pustaka

[1] Adiwidya, B. M . Algoritma Levenshtein Dalam Pendekatan Approximate String Matching. Institut Teknologi Bandung, Bandung. 2009.
[2] Andhika, Fatardhi Rizky. Penerapan String Suggestion dengan Algoritma Levenshtein Distance dan Alternatif Algoritma Lain dalam Aplikasi. Bandung : Institut Teknologi Bandung. 2010.
[3] Priyanto H. Pemrograman Web (HTM / CSS / JavaScript / Power Designer / XAMPP / PHP / CodeIgniter / Jquery). Informatika. 2015.
[4] Achmad S. Pemrograman Web dengan PHP dan MySQL. Universitas Budi Luhur. 2013
[5] Stendy B, S. Aplikasi Web Menggunakan Dreamwever. Yogyakarta: Andi. 2007
[6] Suprianto, D. Buku Pintar Pemrograman PHP. Bandung: OASE Media. 2008
[7] Hanief. Analisis dan Perancangan Sistem Informasi. Yogyakarta: Andi. 2008
[8] Atika, Linda. Analisa dan Perancangan Sistem Informasi. Palembang Universitas Bina Darma.2007