IMPLEMENTASI OTOKOREKSI EJAAN BAHASA INDONESIA PADA KAMUS ISTILAH ELEKTRONIKA

SKRIPSI

Diajukan sebagai salah satu persyaratan untuk memperoleh gelar Sarjana Pendidikan Program Studi Teknik Informatika dan Komputer

oleh

Hilal Aji Wibowo NIM.5302410174

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS NEGERI SEMARANG

2016

IMPLEMENTASI OTOKOREKSI EJAAN BAHASA INDONESIA PADA KAMUS ISTILAH ELEKTRONIKA

SKRIPSI

Diajukan sebagai salah satu persyaratan untuk memperoleh gelar Sarjana Pendidikan Program Studi Teknik Informatika dan Komputer

oleh

Hilal Aji Wibowo NIM.5302410174

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS NEGERI SEMARANG

2016

LAPORAN SELESAI BIMBINGAN

Yang bertanda tangan di bawah ini

Nama : Dr. Hari Wibawanto, M.T

NIP : 196501071991021001

Pangkat / Golongan : IV / A

Sebagai Pembimbing

Melaporkan bahwa penyusunan Skripsi oleh mahasiswa :

Nama : Hilal Aji Wibowo

NIM : 5302410174

Program Studi : Pendidikan Teknik Informatika dan Komputer

Topik : Implementasi Otokoreksi Ejaan Bahasa Indonesia Pada Kamus Istilah Elektronika

Telah selesai dan siap untuk diujikan

Semarang,
Dosen Pembimbing

Dr. Hari Wibawanto, M.T
NIP. 196501071991021001

PERNYATAAN

Saya menyatakan bahwa yang tertulis di dalam skripsi atau tugas akhir benar- benar hasil karya sendiri, bukan jiplakan dari karya tulis orang lain, baik sebagian atau seluruhnya. Pendapat atau temuan orang lain yang terdapat dalam skripsi dirujuk berdasarkan kode etik ilmiah.

Semarang,

Hilal Aji Wibowo

5302410174

PENGESAHAN

Skripsi ini telah dipertahankan dihadapan panitia ujian skripsi Fakultas Teknik Universitas Negeri Semarang pada :

Hari : Rabu

Tanggal : 24 Agustus 2016

Panitia Ujian Skripsi :

Ketua	Sekretaris
Dr.-Ing. Dhidik Prastiyanto, S.T., M.T.	Ir. Ulfah Mediaty Arief, M.T.
NIP. 197805312005011022	NIP. 196605051998022001

Penguji I	Penguji II	Penguji III/Pembimbing
Drs. Suryono, M.T.	Drs. Sugeng Purbawanto, M.T.	Dr. Hari Wibawanto M.T.
NIP. 195503161985031001	NIP. 195703281984031001	NIP. 196501071991021001

Mengetahui

Dekan Fakultas Teknik

Dr. Nur Qudus, M.T.

NIP. 196911301994031001

MOTTO DAN PERSEMBAHAN

Hendaklah kamu semua mengusahakan ilmu pengetahuan itu sebelum dilenyapkan. Lenyapnya ilmu pengetahuan ialah matinya orang-orang yang memberikan atau mengajarkannya. seorang itu tidaklah dilahirkan langsung pandai, jadi ilmu pengetahuan itu pastilah harus dengan belajar.

(Ibnu Mas'ud r.a.)

Skripsi ini dipersembahkan untuk :

- Kedua orang tua yang telah memberikan berbagai dukungan dan motivasi agar menjadi manfaat ilmu yang didapatkan kelas nantinya.
- Sahabat-sahabat yang telah membantu saya dalam mengerjakan skripsi
- Berbagai pihak yang telah membantu pengerjaan tugas akhir skripsi ini
- Lagu - lagu dari Black Sabbath, The Police, Sting, Adele, Asking Alexandria, Coldplay yang selalu menemani mengerjakan skripsi saya.

KATA PENGANTAR

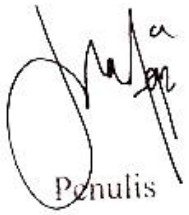
Segala puji bagi Allah SWT, Tuhan semesta alam, yang dengan limpahan rahmat, karunia, serta hidayah-Nya sehingga skripsi yang berjudul **Implementasi Otokoreksi Ejaan Bahasa Indonesia pada Kamus Istilah Elektronika** dapat terselesaikan dengan baik sebagai persyaratan untuk mendapatkan gelar sarjana pendidikan di Universitas Negeri Semarang.

Terselesaikannya penulisan skripsi karena berkat bantuan dari berbagai pihak. Untuk itu diucapkan terima kasih kepada :

1. Bapak Dr. Nur Qudus, M.T. Dekan Fakultas Teknik Universitas Negeri Semarang.
2. Bapak Dr.-Ing. Dhidik Prastiyanto S.T., M.T. Ketua Jurusan Teknik Elektro Universitas Negeri Semarang.
3. Ibu Ir. Ulfah Mediaty Arief, M.T. Koordinator Prodi Pendidikan Teknik Informatika dan Komputer Universitas Negeri Semarang.
4. Bapak Dr. Hari Wibawanto, M.T Dosen pembimbing.
5. Segenap pihak yang terlibat dalam pembuatan proses skripsi ini.

Semoga dengan pembuatan skripsi ini dapat memberikan manfaat sebagaimana yang diharapkan. Amin.

Semarang,



Penulis

ABSTRAK

Wibowo, Hilal Aji. 2016. Implementasi Fitur Otokoreksi Ejaan Bahasa Indonesia pada Kamus Istilah Elektronika. Skripsi, Jurusan Teknik Elektro, Universitas Negeri Semarang. Dr. Hari Wibawanto, M.T.

Kamus saat ini tersedia secara *online* maupun *offline*, pencarian kata secara *online* mengakibatkan proses pencarian yang lama karena basis datanya tersimpan di *server online.* Sedangkan aplikasi yang *offline* masih terdapat beberapa aplikasi yang sedikit menggunakan fitur pencarian tanpa pengoreksi kata maupun otokoreksi. Dengan hal ini perangkat lunak Kamus Istilah Elektronika akan menerapkan fitur otokoreksi dengan algoritma *levenshtein distance* dalam pencarian kata. Tujuan dari penelitian ini adalah untuk mengetahui bagaimana cara mengimplementasikan fitur otokoreksi dan algoritma *levenshtein distance* pada aplikasi Kamus Istilah Elektronika.

Metode pengembangan aplikasi menggunakan metode *waterfall*, yang terdiri dari 5 bagian yaitu definisi kebutuhan, rancangan perangkat lunak dan sistem, implementasi, pengujian serta operasi dan pemeliharaan. Pengujian aplikasi Kamus Istilah Elektronika menggunakan pengujian *black box*, yaitu : pengujian terhadap kemunculan otokoreksi, pengujian terhadap pemberian saran dengan *levenshtein distance.*

Hasil penelitian pada pengujian fitur otokoreksi adalah muncul untuk setiap kata yang dimasukkan dan pada algoritma *levenshtein distance* saran yang akan muncul sesuai dengan kata masukannya. Fitur otokoreksi diimplementasikan pada aplikasi Kamus Istilah Elektronika dengan menambahkan fitur tersebut pada kotak pencarian *edittext*, serta algoritma *levenshtein distance* diimplementasikan pada tombol pencarian.

Kata kunci : *Otokoreksi, Levenshtein Distance, Kamus, Pencarian, Elektronika.*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SKRIP

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Istilah adalah kata atau gabungan kata yang dengan cermat mengungkapkan suatu makna, konsep proses, keadaan, atau sifat yang khas dalam bidang tertentu. Ada dua macam istilah: (1) istilah khusus dan (2) istilah umum. Istilah khusus adalah kata yang pemakaiannya dan maknanya terbatas pada suatu bidang tertentu, misalnya cakar ayam (bangunan), agregat (ekonomi); sedangkan istilah umum ialah kata yang menjadi unsur bahasa umum. Misalnya: ambil alih, daya guna, kecerdasan, dan tepat guna merupakan istilah umum, sedangkan radiator, pedagogi, androgogi, panitera, sekering, dan atom merupakan istilah khusus. Istilah dalam bahasa Indonesia bersumber pada: kosa kata umum bahasa Indonesia, kosa kata bahasa serumpun. dan kosa kata bahasa asing.

Proses pembentukan istilah dilakukan melalui pemadanan atau penerjemahan, misalnya *busway* menjadi jalur bis; penyerapan kosa kata asing, misalnya *camera* menjadi kamera dan gabungan penerjemahan dan penyerapan, misalnya *subdivision* menjadi sub bagian.

Sedangkan kamus merupakan buku acuan yang memuat kata dan ungkapan. Berfungsi untuk membantu mengenal perkataan atau istilah baru. Selain menerangkan maksud kata, kamus juga mungkin mempunyai pedoman sebutan, asal-usul (etimologi) sesuatu perkataan dan juga contoh penggunaan bagi sesuatu

perkataan. Untuk memperjelas kadang kala terdapat juga ilustrasi di dalam kamus.

Biasanya penyusunan kata dan ungkapan disesuaikan dengan urutan abjad berikut dengan keterangan, pemakaian atau terjemaahan. Kamus terdiri dari beberapa jenis, antara lain : Kamus Istilah, Kamus Etimologi, Kamus Tesaurus, Kamus Peribahasa atau Simpulan Bahasa, Kamus Kata Nama Khas, Kamus Terjemahan, Kamus Kolokasi. Dari beberapa jenis kamus tersebut dalam penelitian ini dibuat kamus isitilah pada bidang ilmu elektronika secara *offline*.

Kamus Istilah Elektronika yang akan dibuat dilengkapi fitur otokoreksi sebagai fitur penunjang kecepatan kinerja pencarian istilah kata dalam Kamus Istilah Elektronika. Fitur otokoreksi akan menggunakan algoritma *levenshtein distance.*

1.2 Permasalahan

Berdasarkan latar belakang yang telah diuraikan diatas, konsentrasi permasalahan yang akan diteteliti sebagai berikut :

a. Bagaimana implentasi algoritma *levenshtein distance* dalam aplikasi Kamus Istilah Elektronika ?.
b. Bagaimana efektifitas algoritma *levenshtein distance* dalam aplikasi Kamus Istilah Elektronika ?.

1.3 Batasan Masalah

Berdasarkan permasalahan yang telah dijabarkan guna pengembangan kamus istilah mengurangi kesalahan terhadap penulisan kata-kata, Dalam peneletian ini akan dibatasi berbagai masalah. Hal tersebut adalah :

a. Pengujian dan penerapan algoritma *levenshtein distance.*
b. Efektivitas menemukan target kata yang dituju dengan *levenshtein distance.*
c. Kamus hanya sebagai objek penelitian algoritma *levenshtein distance*.

1.4 Tujuan Penelitian

Berdasarkan masalah yang telah diuraikan pada sub bab sebelumnya maka tujuan dari penelitian ini adalah pemanfaatan algoritma *levenshtein distance* terhadap koreksi kata – kata dalam bahasa indonesia dalam aplikasi Kamus Istilah Elektronika.

1.5 Manfaat Penelitian

Manfaat yang diharapakan dari penelitian yang dilakukan adalah sebagai rujukan atau saran dalam implementasi dalam bentuk kamus istilah elektronika dengan menggunakan algoritma *levenshtein distance*.

1.6 Sistematika Penulisan

Sistematika penulisan unntuk skripsi adalah sebagai berikut :

BAB I PENDAHULUAN

Pada bab ini berisi tentang latar belakang masalah, permasalahan, batas masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

BAB II LANDASAN TEORI

Pada bab ini berisikan pengertian kamus, algoritma *lavenshetein distance*, *visual studio*, data dan basis data, perangkat perancangan sistem, peelitian yang relevan.

BAB III METODE PENELITIAN

Pada bab ini berisi tentang perangkat lunak, penerapan fitur otokoreksi, penerapan algoritma *levenshtein distance*.

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisi tentang hasil perancangan sistem, hasil implementasi fitur otokoreksi, hasil implementasi algoritma *levenshtein distance*, hasil pegujian dan pembahasan,

BAB V KESIMPULAN DAN SARAN

Pada bab ini bersisikan tentang kesimpulan dan saran.

BAB II

LANDASAN TEORI

2.1 Kamus

Kamus sendiri diserap dari bahasa Arab *qamus*, dengan bentuk jamaknya *qawamis*. Kata Arab itu sendiri berasal dari kata Yunani *okeanos* yang berarti lautan. Sejarah kata itu jelas memperlihatkan makna dasar yang terkandung dalam kata kamus, yaitu wadah pengetahuan, khususnya pengetahuan bahasa, yang tidak terhingga dalam dan luasnya. dalam pengertian lain, Kamus adalah buku acuan yang memuat kata dan ungkapan, biasanya disusun menurut abjad beserta penjelasan tentang makna dan pemakainya (Kamus Besar Bahasa Indonesia). Kamus disusun sesuai dengan abjad dari A-Z dengan tujuan untuk memudahkan pengguna kamus dalam mencari istilah yang diinginkannya dengan cepat dan mudah. Kamus memiliki kegunaan untuk memudahkan penggunanya dalam mencari istilah-istilah yang belum dipahami maknanya.

Menurut Dr. Imel Ya'qub (dalam Hakim, 2013:4), macam-macam kamus dibedakan menjadi 8 macam, yaitu :

a. Kamus bahasa

   Merupakan kamus yang secara khusus membahas lafal atau kata-kata dari sebuah bahasa dan dilengkapi dengan pemakaian kata-kata tersebut.

Kamus bahasa hanya membuat satu bahasa, sehingga biasanya pemaknaan kata hanya menyebut sinonim ataupun definisi kata tersebut.

b. Kamus terjemah

Atau disebut juga kamus campuran atau *bilingual* yang memadukan dua bahasa untuk menentukan titik temu makna dari kosakata. Kamus terjemah memuat kata-kata asing yang kemudian dijelaskan satu persatu dengan mencari makna yang sama dan disesuaikan dengan bahasa nasional atau bahasa pemakai kamus.

c. Kamus tematik

Yaitu kamus yang kata-katanya terhimpun di dalam kamus yang disusun secara tematik berdasarkan topik-topik tertentu yang mempunyai makna sebidang.

d. Kamus derivative

Disebut juga dengan istilah kamus etimologis, yaitu sebuah kamus yang membahas asal usul sebuah kata, sehingga kamus derivatif berfungsi untuk menginformasikan asal usul kosakata.

e. Kamus evolutif

Merupakan kamus yang lebih memprioritaskan sejarah perkembangan makna dari sebuah kata, bukan lafalnya. Kamus evolutif memberikan informasi tentang perluasan makna, perubahannya, sebab-sebab perubahan makna dan sebagainya.

f. Kamus spesialis

Yaitu kamus yang menghimpun kata-kata yang ada dalam satu bidang atau disiplin ilmu tertentu. Contohnya yaitu kamus kedokteran, kamus pertanian, kamus musik dan lain sebagainya.

g. Kamus informatif

Yaitu kamus yang mencakup segala hal termasuk sejarah pengguna bahasa, tokoh, dan sebagainya, atau disebut juga dengan ensiklopedia. Di mana menjelaskan sebuah kata yang tidak hanya sekedar membahas makna dan derivasi dari sebuah kata, tapi juga mencakup segala informasi lain diluar makna leksikon, seperti sejarah, biografi, peta, dan lain sebagainya.

h. Kamus *visual*

Merupakan kamus yang menjelaskan makna kata yang lebih menonjolkan gambar dari kata yang dimaksud daripada sebuah istilah yang definitif.

Dari macam-macam jenis kamus tersebut, sebagai pengguna harus memilih secara jelas mana yang lebih dibutuhkan. Dalam menggunakan kamuspun paling tidak hal dasar harus diketahui dalam mencari kata yang diinginkan. Seperti misalnya ketika ingin mencari kata dalam Kamus Besar Bahasa Indonesia. Yang harus diperhatikan adalah, lihat dulu apakah kata tersebut terdiri dari kata dasar atau tidak. Misalkan mencari kata "kehukuman", tentu pengguna tidak akan menemukan kata tersebut dalam formasi huruf "k". Karena kata "kehukuman"

mempunyai kata dasar "hukum" dan memiliki entri atau lema "penghukum", "penghukuman", "hukuman" dan "kehukuman" (Wiyanto, 2005).

Kamus digital adalah kamus yang lebih mengutamakan pada fasilitas pengolah kata elektronis, yaitu sebuah fasilitas yang memungkinkan aplikasi pengolah kata memeriksa ejaan dari dokumen yang diketik (Rina Rizky Harfatiani, 2007 : 37).

Pengguna kamus elektronis atau kamus digital dalam aplikasi pemrosesan teks merupakan hal yang tidak dapat dihindarkan. Kamus merupakan basis pemeriksaan, basis pengetahuan, bahkan sebagai basis penyelidikan (Rina Rizky Harfatiani, 2007 : 37).

Dari beberapa informasi pengertian tersebut dapat ditarik kesimpulan bahwa kamus digital adalah aplikasi kata yang berupa digital, memeriksa dan menemukan data kata yang sudah disimpan dalam arsipnya.

2.2 Algoritma *Levenshetein Distance*

Dalam teori informasi, *levenshtein distance* dua *string* adalah jumlah minimal operasi yang dibutuhkan untuk mengubah suatu *string* ke *string* yang lain, di mana operasi-operasi tersebut adalah operasi penyisipan, penghapusan, atau penggantian sebuah karakter. Algoritma ini dinamakan berdasarkan dari nama penemunya yaitu Vladimir Levenshtein, yang menemukan algoritma tersebut pada tahun 1965. *Levenshtein distance* dirujuk dengan menggunakan kata jarak

saja agar lebih singkat. Algoritma dasar penentuan jarak dua *string* ini dapat dibentuk melalui hubungan rekursif.

Metode *levenshtein distance* dapat digunakan untuk melakukan perhitungan beda jarak antara dua *string*. Dalam istilah umum sering disebut *edit distance*. Jarak atau *distance* adalah jumlah minimum dari operasi penyisipan, penghapusan atau penggantian yang dibutuhkan untuk mengubah *string* asal menjadi *string* target. Sebagai contoh kata "robot" (*string* target) dan kata "robat" (*string* asal) memiliki jarak 1 karena diperlukan satu operasi untuk mengubah kata "robat" menjadi kata "robot".

Dalam pengimplementasiaannya, *levenshtein distance* digunakan dalam aplikasi *spell checker*, *T9 Predict*, Kamus, *Search Engine* sederhana, maupun pendeteksi plagiat.

Dalam algoritma *levenshtein distance* terdapat 3 macam operasi utama yang dapat dilakukan, operasi tersebut adalah :

a. Operasi pengubahan karakter

Operasi pengubahan karakter adalah operasi untuk menukar satu karakter dengan karakter lain sesuai dengan karakter target. Seperti contoh pengubahan dari kata "voll" menjadi kata "volt". Dimana karakter asal "l" diubah menjadi karakter target yaitu karakter "t".

b. Operasi penambahan karakter

Operasi ini sesuai dengan penyebutannya dimana akan ditambah satu karakter sesuai dengan karakter target dalam satu kata. Seperti contoh kata targetnya adalah "ampere" dibandingkan dengan kata asal "amper". Operasi penambahan secara teoritis akan ditambahkan karakter "e" karena kata asal kekurangan satu karakter yaitu karakter huruf "e" untuk memenuhi kata target. Operasi penambahan karakter dapat terjadi di depan, di tengah, maupun di akhir dari susunan kata.

c. Operasi penghapusan karakter

Operasi penghapusan karakter merupakan operasi untuk menghilangkan satu karakter sesuai dengan karakter target yang dibandingkan. Misal kata "calibrate" dengan kata "callibrate" kata target adalah "calibrate". Penghilangan satu karakter "l" merupakan operasi penghapusan karakter.

Dalam kasus seperti kata yang dibandingkan adalah "latency" dan "lsteny", secara perhitungan memiliki nilai perhitungan *edit distance* 3. Dengan tiga operasi yaitu 1) menghapus karakter "s" pada "lsteny" sehingga menjadi "lteny" 2) menambahkan karakter "a" pada kata yang telah dihapus yaitu dari "lteny" sehingga menjadi "lateny" dan 3) menambahkan karakter "c" dari kata yang telah ditambahkan yaitu "lateny" sehingga menjadi "latency".

Proses perubahan tersebut ada dua macam proses yaitu penambahan karakter "a" dan karakter " c" serta penghapusan karakter "s". Algoritma ini berjalan mulai dari pojok kiri aras sebuah *array* dua dimensi yang telah diisi sejumlah karakter dalam kata salah dan kata target dan diberikan nilai *cost*. Nilai *cost* pada ujung kanan bawah seperti gambar 2.1 menjadi nilai *edit* ***distance*** yang menggambarkan jumlah perbedaan dua kata. Gambar 2.1 merupakan contoh dari perhitungan dengan tabel matriks *levenshtein distance* yaitu dengan membandingkan kata "latency" dengan "lsteny".

		L	S	T	E	N	Y	0
	0	1	2	3	4	5	6	7
L	1	0	1	2	3	4	5	6
A	2	1	1	2	3	4	5	6
T	3	2	2	1	2	3	4	5
E	4	3	3	2	1	2	3	4
N	5	4	4	3	2	1	2	3
C	6	5	5	4	3	2	2	3
Y	7	6	6	5	4	3	2	3

Gambar 2.1 Tabel Matriks Perhitungan *Levenshtein distance*

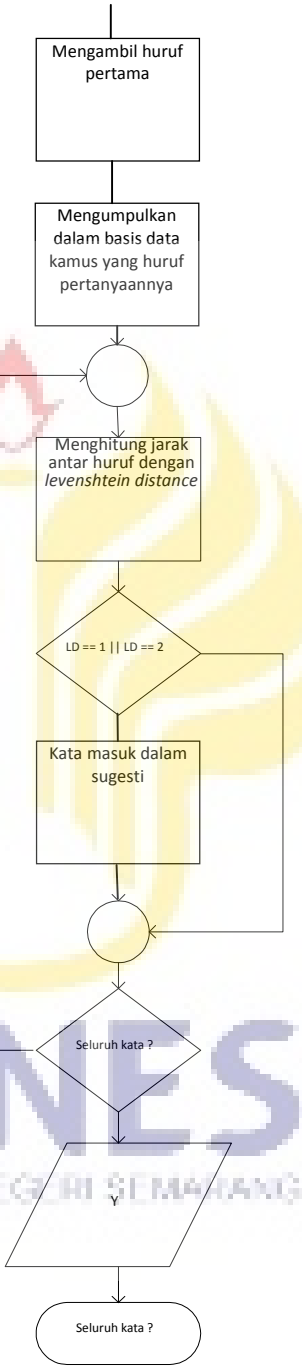
Pada kasus lain dari perhitungan *levenshtein distance* dengan menggunakan tabel *string* "volt" dan "vlot", dapat dilihat pada gambar 2.2 dibawah ini :

		V	L	O	T
	0	1	2	3	4
V	1	0	1	2	3
O	2	1	1	1	2
L	3	2	1	2	2
T	4	3	2	2	2

Gambar 2.2 Tabel Matriks Perhitungan *Levenshtein distance*

Dari gambar 2.2 diatas dapat disimpulkan bahwa hasil perhitungan *edit distance* dalam perbandingan kata "volt" dan "vlot" adalah bernilai 2. dengan operasi pengubahan karakter "l" dan "o".

Proses berjalannya algoritma *levenshtein distance* ini dimulai dari adanya kata salah yang dimasukkan oleh pengguna, kemudian diambil karakter pertama dari masukan di kotak pencarian. Setelah itu kata di dalam basis data yang karakter atau huruf pertamanya sama dikumpulkan, untuk kemudian dihitung jaraknya. Lalu dibandingkan apakah kata asal (masukan kata oleh pengguna) dan kata target (kata dalam basis data) mempunyai kesamaan atau tidak. Jika sama, maka kata tersebut akan masuk ke dalam saran yang akan dimunculkan. Namun jika berbeda maka akan kembali lagi ke langkah awal yaitu mencari jarak dengan algoritma *levenshtein distance*. Namun jika yang muncul adalah keseluruhan kata maka akan ditampilkan saran atau prediksi, disini proses akan berhenti. Untuk lebih jelas dapat melihat gambar 2.3



Gambar 2.3 *Flowchart* dari Algoritma *Levenhstein Distance* (Dwitiyastuti, 2003)

Berikut ini merupakan *pseudocode* dari algoritma *levenshtein distance* menurut (Ilmy, 2006)

```
functionlevDis (s1 : string, s2 : string) :
integer kamus i, j, cost : integer
m : array[0 .. s1.length, 0 .. s2.length] of integer

algoritma

for i <- 0 to s1.length do
for j <- 0 to s2.length do

if i = 0 then
m[i,j] <- j {perbandingan dengan kosong}
else
if j = 0 then
m[i,j] <- i {perbandingan dengan kosong}
else{implementasi pemrograman dinamis}

if s1[i] = s2[j] then
cost <- 0    else   cost <- 1
m[i,j] = minimum (m[i-1, j-1] + cost, {substitusi}
m[i-1,j] + 1, {penghapusan}
m[i ,j-1] + 1, {penambahan}   )
Return m[s1.length, s2.length]
```

Setelah didapat nilai *edit distance* dari dua kata yang telah dibandingkan, maka selanjutnya adalah menghitung nilai kemiripan pada perbandingan kata tersebut. Berikut ini merupakan rumus untuk menghitung kemiripan kata menurut Abraham (2014:227) :

$$Sim = 1 - \frac{d}{Max\ of\ 2\ Strings}$$

Keterangan

Sim : Similarity

d : Nilai *Edit distance*

Max of 2 Strings : Jumlah Maksimal dari kedua kata

2.3 Visual Basic

Visual Basic (Beginners All-Puspose Symbolic Instruction Code) merupakan sebuah bahasa pemrograman yang dapat digunakan untuk membuat suatu aplikasi dalam Microsoft Windows. *Visual Basic* menggunakan metode *Graphical User Interface (GUI)* dalam pembuatan program aplikasi *(Project).* Istilah *visual* mengacu pada metode pembuatan tampilan program *(interface)* Atau objek pemrograman yang biasa dilakukan secara langsung dan terlihat oleh programmer. Dalam *Visual Basic,* pembuatan program aplikasi harus dikerjakan dalam sebuah *project*. Sebuah *Project* dapat terdiri dari *File Project (.vbp), File Form (.frm), File Data Binary (.frx), Modul Class (.cls), Modul Standar (.bas), dan file resource tunggal (.res).* Bahasa yang digunakan adalah bahasa Basic yang sangat popular pada era sistem operasi *DOS.*

2.4 Data

Data dapat didefinisikan sebagai deskripsi dari suatu kejadian yang kita hadapi (Al- Bahra Bin Ladjamudin, 2005). Data dapat berupa catatan-catatan dalam kertas, buku atau tersimpan sebagai *file* di dalam basis data.

Menurut (Wahyudi, Bambang. 2004) kata data diambil dari bahasa Inggris yang berasal dari bahasa Yunani datum yang berarti fakta. Bentuk jamak dari *datum* adalah *data*. Jadi, data adalah suatu nilai mentah yang tidak memiliki arti apa-apa apabila dia berdiri sendiri. Data juga dapat diartikan sebagai deskripsi

tentang bena, kejadian, aktivitas dan transaksi yang tidak mempunyai makna atau tidak berpengaruh secara langsung kepada pemakai.

Data merupakan kumpulan dari angka-angka maupun karakter-karakter yang tidak memiliki arti. Karakter (angka, abjad, simbol) adalah sekelompok kecil *bit* yang pengaturannya memberikan arti tertentu dan *bit* merupakan satuan data yang terkecil dalam proses komputer yaitu terdiri dari angka nol atau satu. Data dapat diolah sehingga menghasilkan informasi (Widodo, P.D.. 2003).

Proses pengolahan data terbagi menjadi tiga tahapan, yang disebut dengan siklus pengolahan data *(Data Processing Cycle)* yaitu:

a. Tahapan *Input*

   Yaitu dilakukan proses pemasukan data kedalam komputer lewat *media input (input devices).*

b. Tahapan *Processing*

   Yaitu dilakukan proses pengolahan data yang sudah dimasukkan yang dilakukan oleh alat pemroses *(Process devices)* yang dapat berupa proses perhitungan, perbandingan, pengendalian dan pencarian di *storage.*

c. Tahapan *Output*

   Yaitu dilakukan proses menghasilkan *output* dari hasil pengolahan data ke alat *output (Output Devices)* yaitu berupa informasi.

2.5 Basis Data

Basis data adalah kumpulan *file-file* yang saling berelasi, relasi tersebut biasa ditunjukkan dengan kunci dari tiap *file* yang ada. Satu basis data menunjukkan satu kumpulan data yang dipakai dalam satu lingkup perusahaan, instansi.

Untuk membuat dan mengakses dapat digunakan *DAO (Data Access Object), ADO (ActiveX Data Object)* atau *File IO (File Input Output).* Aplikasi *Visual Basic* dapat menangani bermacam-macam ekstensi basis data, yaitu ekstensi dari *platform* Microsoft Access, Microsoft Excel, dBAse, FoxPro, Paradox, ODBC, dan *File Text*. Basis daata yang akan digunakan dalam pembuatan Kamus Istilah Elektronika adalah *File* teks.

Penyusunan suatu basis data digunakan untuk mengatasi masalah-masalah pada penyusunan data yaitu:

a. Redudansi dan inkonsistensi data
b. Kesulitan pengaksesan data
c. Isolasi data untuk standarisasi
d. *Multiple user* (Banyak Pemakai)
e. Masalah keamanan (security)
f. Masalah Integrasi (kesatuan)
g. Masalah *data independence* (kebebasan data)

Kegunaan utama sistem basis data adalah agar pengguna mampu menyusun suatu pandangan abstraksi dari data. Bayangan mengenai data tidak lagi memperhatikan kondisi sesungguhnya bagaimana satu data masuk ke basis data, disimpan dalam *disk disektor* mana, tetapi menyangkut secara menyeluruh bagaimana data tersebut dapat diabstraksikan atau digambarkan menyerupai kondisi yang dihadapi pemakai sehari-hari.

Sistem yang sesungguhnya tentang teknis bagaimana data disimpan dan dipelihara seakan akan disembunyikan kerumitannya dan kemudian diungkapkan dalam bahasa dan gambar yang mudah dimengerti orang awam.

2.6 Perangkat Perancangan Sistem

Dalam pembuatan suat aplikasi atau *software* dibutuhkan cara untuk memperesentasikan aplikasi secara keseluruhan namun mudah dipahami. Cara yang bisa digunakan adalah dengan sistem *flowchart, flowmap,* membuat *UML (Unified Modeling Language)* dan lain sebagainya. Berikut ini merupakan perangkat perancangan sistem yang digunakan untuk mempresentasikan aplikasi ini secara lebih sederhana :

1. *Flowchart*

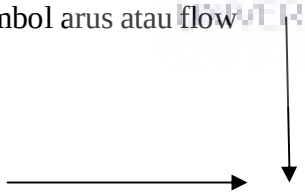
   Flowchart merupakan suatu kumpulan atau aliran bagan yang menggambarkan terjadinya proses prosedur atau program. Jadi dalam suatu *flowchart* yang dibuat, bisa menggambarkan bagaimana suatu program itu berjalan. Sehingga jika penjelasan secara deskripsi masih kurang dipahami

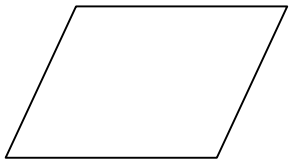
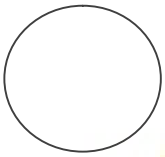
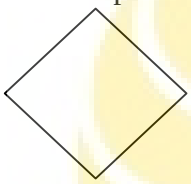
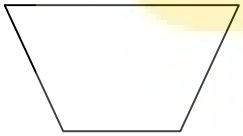
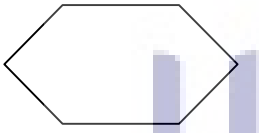
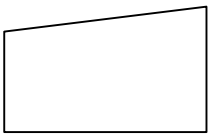
oleh orang lain, maka ada bagan atau *flowchart* yang bisa mudah dipahami karena tersusun lebih sederhana dan singkat. Untuk memulai membuat *flowchart*, berikut ini merupakan hal-hal yang harus diperhatikan dalam proses pembuatannya :

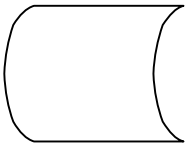
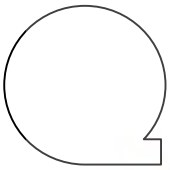
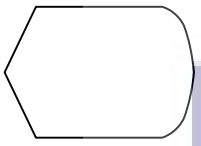
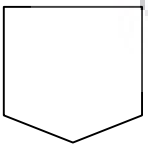
a. *Flowchart* dimulai dari atas kebawah dan dari kiri ke kanan.
b. Proses yang digambarkan harus dijelaskan secara jelas dan dimengerti oleh pembaca.
c. Kapan aktivitas dimulai dan berakhir harus ditentukan dengan jelas.
d. Harus dimulai dari urutan yang benar, jangan sampai terbalik-balik.
e. Gunakan simbol *flowchart* yang standar, dan sesuai dengan jalannya proses program.

Dalam proses pembuatan *flowchart*, terdapat beberapa komponen yang harus diketahui oleh pengguna ketika akan membuatnya. Dan berikut ini merupakan simbol-simbol yang digunakan dalam pembuatan *flowchart* beserta deskripsinya :

Tabel 2.1 Simbol dan definisi *Flowchart*

Simbol	**Definisi**
Simbol arus atau flow 	Menyatakan jalannya arus suatu proses aplikasi berjalan
Process 	Menyatakan suatu proses yang sedang dilakukan oleh program

Data 	Menyatakan proses *masukan* atau *output* dari suatu data
Connector atau penghubung 	Menyatakan sambungan dari proses satu ke proses lainnya namun dalam halaman yang sama.
Decision atau keputusan 	Menyatakan suatu kondisi yang menghasilkan jawaban iya atau tidak.
Manual 	Menyatakan suatu proses yang tidak dilakukan oleh komputer.
Preparation 	Menyatakan tempat sebagai nilai awal variabel diletakkan.
Keying operation 	Menyatakan segala jenis operasi yang diproses dengan menggunakan *keyboard.*
Manual masukan 	Menyatakan bahwa masukan dalam suatu data dilakukan secara manual dengan menggunakan *keyboard*.

Disk storage 	Menyatakan bahwa *masukan* berasal dari *disk* atau *output* yang ada disimpan di *disk*.
Magnetic tape 	Menyatakan bahwa *masukan* yang berasal dari pita magnetis atau *output* yang disimpan ke pita magnetis.
Punched card 	Menyatakan bahwa *masukan* yang ada berasal dari kartu atau *output* yang berasal dari kartu.
Dokumen 	Menyatakan bahwa hasil *output* dicetak dalam bentuk dokumen.
Display 	Menyatakan bahwa *output* yang dicetak akan muncul dalam layar *monitor*.
Offline connector 	Menyatakan bahwa sambungan dari suatu proses ke proses lainnya namun dalam halaman yang berbeda.

2. *UML (Unified Moduling Language)*

UML merupakan suatu kumpulan diagram maupun simbol yang digunakan untuk mempresentasikan atau memodelkan suatu *software* atau perangkat lunak. Dengan adanya UML ini dapat diketahui bagaimana dalam membuat suatu model *software* yang akan dikerjakan, sehingga menjadi jelas dan terarah. Dalam UML ini terdapat 8 diagram, berikut ini diantaranya :

a. *Use case diagram*

Use case diagram merupakan pemodelan dengan menggunakan perspektif pengguna dari suatu sistem, yang terdiri atas diagram *use case* dan aktor. Di mana aktor merupakan orang yang akan menjalankan atau berinteraksi dengan sistem aplikasi.

b. *Class diagram*

Class diagram merupakan diagram yang memiliki *class* dan menggambarkan hubungan antar *class* tersebut dengan sistem. Di mana *class* ini digambarkan dengan membentuk suatu kotak menjadi 3 bagian.

c. *Package diagram*

Package diagram digunakan untuk mengatur diagram *class* yang kompleks dan banyak sehingga dibuat *package* agar tertata dengan baik dalam pengelompokkan kelas.

d. *Activity diagram*

Activity diagram merupakan diagram yang lebih menekankan pada aktivitas yang terjadi pada proses aplikasi sistem tersebut berjalan dan saling terhubung atau bergantung satu sama lainnya.

e. *Deployment diagram*

Deployment diagram adalah diagram yang menerangkan konfigurasi fisik *software* dan *hardware* dari aplikasi yang akan dijalankan.

f. *Statechart diagram*

Statechart diagram merupakan diagram yang menunjukkan keadaan obyek dan proses yang menyebabkan perubahan pada keadaannya.

g. *Collaboration diagram*

Collaboration diagram membawa informasi yang sama dengan diagram *sequence*, akan tetapi lebih memfokuskan kegiatan obyek dari waktu pesan tersebut dikirimkan.

h. *Sequence diagram*

Sequence diagram merupakan suatu diagram yang menjelaskan bagaimana sistem itu berjalan atau operasi itu dilakukan, dan diagram ini diatur berdasarkan waktu.

2.7 Penelitian yang Relevan

Penelitain yang dilakukan oleh Pyriana, Dewi Rokhmah, dkk. (2013) tentang Program Aplikasi Editor Kata Bahasa Indonesia Menggunakan Metode *Approximate String Matching* Dengan Algoritma *Levenshtein distance* Berbasis Java. Menyimpulkan bahwa dalam pengujian cek editor kata memiliki akurasi 77,35% terhadap banyak variasi kata salah. Pada menu cek struktur kalimat dengan 10 kalimat tunggal, akurasi yang dihasilkan 100%. Sedangkan waktu yang dibutuhkan untuk eksekusi berbanding lurus dengan jumlah input kata. Semakin banyak kata yang dimasukkan, semakin banyak pula waktu yang dibutuhkan untuk eksekusi. Sedangkan jumlah banyaknya kesalahan ejaan kata pada suatu teks tidak berpengaruh pada waktu eksekusi.

Penelitian yang dilakukan oleh Yoke Yie., dkk (20130 Ishak mengenai Pendeteksian Sistem *Email Hoax* Menggunakan Metode *Levenshtein distance* menyimpulkan bahwa hasil dari penelitian tersebut adalah sistem sudah dapat memberikan prediksi positif dengan nilai 0.96, namun masih belum dapat mengidentifikasi yang mana email asli. Kemudian penelitian yang berjudul Penggunaan Algoritma *Edit distance* Untuk Mencari Kata Kunci di

Cloud Environment yang disusun oleh Abraham, Riya Mary (2014) menyimpulkan bahwa dengan menggunakan algoritma *levenshtein distance* pada pencarian di *cloud* dapat memberikan pencarian yang lebih banyak dibandingkan dengan menggunakan pencarian tradisional.

Lalu penelitian yang dilakukan oleh Yao, Zhiqiang (2013) mengenai Implementasi Fitur *Autocomplete* pada *Textbox* yang Berdasar pada *Ajax* dan *Web Service* menyimpulkan bahwa dengan menggunakan fitur *autocomplete* ini masukkan dari pengguna menjadi lebih efisien dan mengalami peningkatan. Selain itu pengejaan yang salah bisa dihindari dan interaksi manusia pada komputer jadi menguat. Selain itu pada penelitian yang berjudul Pengadaptasian *Levenshtein distance* pada Koreksi Pengejaan Kontekstual yang disusun oleh Lhoussain, Aouragh Si (2013) menyimpulkan bahwa dengan mengkombinasikan model *bi-grams* dengan *levenshtein distance* dapat meningkatkan kepuasan dalam solusi metode tersebut.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Hasil dari pengembangan aplikasi Kamus Istilah Elektronika yang telah dilakukan dapat ditarik kesimpulan :

1. Pengunaan fitur otokoreksi dalam pencarian kata istilah dapat memudahkan pengguna dalam mencari istilah dan memudahkan hasil pencarian istilah terdekat apabila terjadi kesalahan dalam penulisan masukan kata yang akan dicari.
2. Fitur otokoreksi yang diimplementasikan dalam Kamus Istilah Elektronika sangat efektif untuk digunakan.

5.2 Saran

Saran untuk pengembangan aplikasi Kamus Istilah Elektronika adalah :

1. Masih perlu penambahan basis data istilah elektronika.
2. Kurangnya pengujian kata-kata istilah yang lebih kompleks untuk menguji ketepatan dari saran kata.

DAFTAR PUSTAKA

Abraham, Riya Mary. 2014. *Use of Edit distance Algorithm to Search A Keyword In Cloud Environment.* International Journal of Database Theory And Application. (7) 6 : 223-232.

Chen, Yoke Yie., dkk. 2014. *E-Mail Hoax Detection System Using Levenshtein distance Method.* Journal of Computer. (9) 2 : 445-446.

Dwitiyastuti, Rachmania Nur,. dkk. 2013. *Pengoreksi Kesalahan Ejaan Bahasa Indonesia Menggunakan Metode Levenshtein distance.* http://www.eletronika.studentjournal.ub.ac.id/. Diakses pada 10 Februari 2015.

Hakim, Ahmad Luqman, Ferdian Rikza. 2013. *Sistematika Penulisan Kamus Berdasarkan Entri, Jumlah Bahasa, dan Metode Masa/Periode.* http://www.fai.uad.ac.id/. Diakses pada 3 Februari 2015

Harfatiani, Rina Rizky. 2007. *Teknik Riset Operasi*. Surabaya: Kartika.

Ilmy, Muhammad Bahari., dkk. 2006. *Penerapan Algortima Levenshtein distance untuk Mengoreksi Kesalahan Pengejaan pada Editor Teks.* http://www.informatika.stei.itb.ac.id/. Diakses pada 10 Februari 2015.

Ladjamuddin, Al-Bahra bin. 2005. *Analisis dan Desain Sistem Informasi.* Yogyakarta: Graha Ilmu.

Lhoussain, Aouragh Si. 2005. *Adapting The Levenshtein distance to Contextual Spelling and Correction.* International Journal of Computer Science and Application. (12) 1 : 127-133.

Pressman, Roger S. 2002. *Rekayasa Perangkat Lunak: Pendekatan Praktisi (Buku 1).* Yogyakarta: Andi.

Pyriana, Dewi Rokhmah., dkk. 2013. *Program Aplikasi Editor Kata Bahasa Indonesia Menggunakan Metode Approximate String Matching dengan Algoritma Levenshtein distance Berbasis Java.* http://www.filkom.ub.ac.id/. Diakses pada 10 Februari 2015.

Rosa A, S dan M Salahuddin. 2011. *Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek).* Bandung: Informatika.

Sugiyono. 2012. *Metode Penelitian Kuantitatif Kualitatif dan R&D.* Bandung: Alfabeta.

Wahyudi, Bambang. 2003. *Pengantar Struktur Data dan Algoritma.* Yogyakarta: Andi.

Widodo, P.D. 2003. *Kamus Istilah Internet dan Komputer.* Jombang: Lintas Media.

Wiyanto, Asul., dkk. 2005. *Mampu Berbahasa Indonesia.* Jakarta: Grasindo.

Yao, Zhiqiang. 2013. *Implementation of The Autocomplete feature of The Textbox Based on Ajax and Web Service.* Journal of Computer (8) 9 : 2199-2201.