Implementasi Fitur *Autocomplete* dan Algoritma *Levenshtein Distance* untuk Meningkatkan Efektivitas Pencarian Kata di Kamus Besar Bahasa Indonesia (KBBI)

SKRIPSI

diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana pendidikan Prodi Pendidikan Teknik Informatika dan Komputer

oleh

Khairun Nisa Meiah Ngafidin
5302411066

PENDIDIKAN TEKNIK INFORMATIKA DAN KOMPUTER

FAKULTAS TEKNIK

UNIVERSITAS NEGERI SEMARANG

2015

LAPORAN SELESAI BIMBINGAN

Yang bertanda tangan dibawah ini

Nama	: Dr. Hari Wibawanto, M.T
NIP	: 196501071991021001
Pangkat / Golongan	: IV / A
Sebagai Pembimbing	

Melaporkan bahwa penyusunan Skripsi oleh mahasiswa :

Nama	: Khairun Nisa Meiah Ngafidin
NIM	: 5302411066
Program Studi	: Pendidikan Teknik Informatika dan Komputer
Topik	: Implementasi Fitur Autocomplete dan Algoritma Levenshtein Distance untuk Meningkatkan Efektivitas Pencarian Kata di Kamus Besar Bahasa Indonesia (KBBI)

Telah selesai dan siap untuk diujikan

UNNES
UNIVERSITAS NEGERI SEMARANG

Semarang, Agustus 2015

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, Oktober 2015

Khairun Nisa Meiah Ngafidin

5302411066

UNNES
UNIVERSITAS NEGERI SEMARANG

iii

PENGESAHAN

Skripsi ini telah dipertahankan dihadapan panitia ujian skripsi FT UNNES pada

Hari : Kamis

Tanggal : 1 Oktober 2015

Panitia Ujian Skripsi :

Ketua

Drs. Subyono, M.T
NIP. 195503161985031001

Sekretaris

Feddy Setio Pribadi, S.Pd.,M.T
NIP.197808222003121002

Penguji I

Dr.Ir.Subiyanto,S.T, M.T
NIP.197411232005011001

Penguji II

Anggraini Mulwinda, S.T, M.Eng
NIP. 197812262005012002

Penguji III/Pembimbing

Dr. Hari Wibawanto, M.T
NIP.196501071991021001

Mengetahui
Dekan Fakultas Teknik

Dr. Nur Qudus M.T
NIP. 196911301994031001

MOTTO DAN PERSEMBAHAN

Janganlah engkau merasa sombong terhadap ilmu, karena ilmu adalah lembah-lembah. Yang manapun engkau tempuh, dia akan mengalahkanmu sebelum engkau mencapainya. Akan tetapi ambillah ilmu itu bersamaan dengan perjalanan siang dan malam. Dan janganlah engkau mengambil ilmu sekaligus, akan hilang pula sekaligus. Akan tetapi ambillah ilmu sedikit demi sedikit, bersamaan dengan perjalanan siang dan malam.

(Imam Az-Zuhri)

Skripsi ini penulis persembahkan untuk :

- Kedua orang tuaku, yang telah memberikan berbagai macam dorongan dan motivasi agar menjadi sukses dan lancar dalam pengerjaan skripsi ini. Juga perhatian yang tiada habisnya.
- Adikku yang memberikan semangatnya agar lulus dengan baik.
- Sahabat-sahabat, teman kos yang banyak membantuku dan ada ditiap keadaan.
- Semua orang yang kukenal yang telah memberikanku dorongan dan semangat yang luar biasa.

UNNES
UNIVERSITAS NEGERI SEMARANG

KATA PENGANTAR

Segala puji bagi Allah SWT, Tuhan semesta alam, yang dengan limpahan rahmat-Nya, karunia, serta hidayah-Nya sehingga skripsi ini yang berjudul **Implementasi Fitur *Autocomplete* dan Algoritma *Levenshtein Distance* untuk Meningkatkan Efektivitas Pencarian Kata di Kamus Besar Bahasa Indonesia** ini dapat terselesaikan dengan baik sebagai persyaratan untuk mendapatkan gelar sarjana pendidikan di Unnes tercinta.

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

1. Bapak Drs. Suryono, M.T selaku ketua Jurusan Teknik Elektro.
2. Bapak Feddy Setio Pribadi S.Pd, M.T selaku ketua Prodi Pendidikan Teknik Informatika dan Komputer Universitas Negeri Semarang.
3. Bapak Dr. Hari Wibawanto, M.T selaku dosen pembimbing.
4. Segenap pihak yang terlibat dalam pembuatan proses skripsi ini.

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

UNNES
UNIVERSITAS NEGERI SEMARANG

Semarang, Agustus 2015

Penulis

ABSTRAK

Ngafidin, Khairun Nisa Meiah. 2015. Implementasi Fitur *Autocomplete* dan Algoritma *Levenshtein Distance* untuk Meningkatkan Efektivitas Pencarian Kata di Kamus Besar Bahasa Indonesia (KBBI). Skripsi, Jurusan Teknik Elektro, Program Studi Pendidikan Teknik Informatika dan Komputer, Fakultas Teknik, Universitas Negeri Semarang. Dr. Hari Wibawanto, M.T.

Kamus Besar Bahasa Indonesia 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 minim fitur pencarian tanpa pengoreksi kata maupun tanpa *autocomplete*. Dengan ini aplikasi KBBI akan menerapkan fitur *autocomplete* dan algoritma *levenshtein distance* pada proses pencarian katanya. Tujuan dari penelitian ini adalah untuk mengetahui bagaimana cara mengimplementasikan fitur *autocomplete* dan algoritma *levenshtein distance* pada aplikasi KBBI, serta untuk mengetahui efektifivitas penggunaannya.

Metode pengembangan *software* ini menggunakan metode *waterfall*, yang terdiri dari 5 bagian yaitu definisi kebutuhan, rancangan perangkat lunak dan sistem, implementasi, pengujian, serta operasi dan pemeliharaan. Pengujian aplikasi ini menggunakan metode *black box*, yaitu : pengujian terhadap kemunculan *autocomplete*, pengujian terhadap pemberian saran dengan *levenshtein distance*, serta pengujian terhadap keefektifan fitur *autocomplete* dan algoritma *levenshtein distance* sebagai pengoreksi kata.

Hasil penelitian pada pengujian fitur *autocomplete* adalah muncul untuk setiap kata yang dimasukkan dan pada algoritma *levenhtein distance* saran yang muncul sesuai dengan kata masukkannya. Hasil pengujian efektivitas *autocomplete* adalah sebesar 84.615 % yang berarti sangat efektif, dan *levenshtein distance* sebesar 76.04 % yang berarti efektif untuk digunakan di aplikasi KBBI. Fitur *autocomplete* diimplementasikan pada aplikasi KBBI dengan menambahkan fitur tersebut pada kotak pencarian *edittext*, serta algoritma *levenshtein distance* diimplementasikan pada tombol pencarian.

Kata kunci : *Autocomplete, Levenshtein distance, Kamus, Pencarian, KBBI*.

DAFTAR ISI

LAPORAN SELESAI BIMBINGAN ii

PERNYATAAN iii

PENGESAHAN iv

MOTTO DAN PERSEMBAHAN v

KATA PENGANTAR vi

ABSTRAK vii

DAFTAR ISI viii

DAFTAR TABEL xiii

DAFTAR LAMPIRAN xv

BAB I PENDAHULUAN 1

1.1 Latar Belakang 1

1.2 Rumusan Masalah 7

1.3 Tujuan Penelitian 7

1.4 Manfaat Penelitian 7

1.5 Batasan Masalah 8

1.6 Definisi Operasional Judul Penelitian 9

BAB II LANDASAN TEORI 11

2.1 Kamus 11

2.2 KBBI 14

2.3 KBBI Digital 16

2.4 Fitur *Autocomplete* 18

2.5 Algoritma Levenshtein Distance 21

2.6 Android 27

2.7 Perangkat Perancangan Sistem 28

2.9 Penelitian yang Relevan 33

BAB III METODE PENELITIAN 35

3.1 Metode Pengembangan Software 35

UNNES
UNIVERSITAS NEGERI SEMARANG

3.1.1 Definisi Kebutuhan 36
3.1.2 Perancangan Perangkat Lunak dan Sistem 37
3.1.3 Implementasi 54
3.1.4 Pengujian 56
3.1.5 Pengoperasian dan Pemeliharaan 60
3.2 Teknik Pengumpulan Data 60
3.3 Penerapan Fitur Autocomplete 63
3.4 Penerapan Algoritma Levenshtein Distance 64
BAB IV HASIL DAN PEMBAHASAN 66
4.1 Implementasi Pengkodean Fitur Autocomplete 66
4.2 Implementasi Pengkodean Algoritma Levenshtein Distance 68
4.3 Hasil Tampilan Aplikasi KBBI 72
4.4 Pengujian 83
4.5 Hasil Pengujian 88
4.3.1 Hasil Uji Fungsi 88
4.3.2 Hasil Pengujian Fitur *Autocomplete* 89
4.3.3 Hasil Pengujian Algoritma *Levenshtein distance* 91
4.6 Pengujian Efektivitas 101
4.7 Pembahasan 103
BAB V KESIMPULAN DAN SARAN 108
5.1 Kesimpulan 108
5.2 Saran 108
DAFTAR PUSTAKA 109
LAMPIRAN 112

UNNES
UNIVERSITAS NEGERI SEMARANG

DAFTAR GAMBAR

Gambar 1. 1 KBBI daring tanpa fitur *autocomplete* dan KBBI dengan *autocomplete* di luar jaringan. 4

Gambar 2. 1 Contoh KBBI Digital Berupa di badanbahasa.kemdikbud.go.id 17

Gambar 2. 2 Contoh KBBI Digital dalam Bentuk Luar Jaringan 17

Gambar 2. 3 Contoh *Screenshot* Aplikasi KBBI Berbasis Android 18

Gambar 2. 4 Contoh dari Proses *Autocomplete* (Yao, 2013) 19

Gambar 2. 5 Flowchart dari Proses Algoritma *Levenshtein Distance* (Dwitiyastuti, t.t) 25

Gambar 2. 6 Pseudocode dari Algoritma *Levenshtein distance* 26

Gambar 3. 1 Flowchart Tahapan Pembuatan Aplikasi KBBI 35

Gambar 3. 2 Use Case Diagram 40

Gambar 3. 3 Activity Diagram Proses *Autocomplete* 41

Gambar 3. 4 Activity Diagram Membuka Menu Tentang Aplikasi 42

Gambar 3. 5 Activity Diagram Menutup Aplikasi 42

Gambar 3. 6 Use Case Diagram dari Menu Login Admin 43

Gambar 3. 7 Sequence Diagram dari Menu Tentang Aplikasi 44

Gambar 3. 8 Sequence Diagram Menu Keluar 45

Gambar 3. 9 Sequence Diagram Menu Lihat Data 45

Gambar 3.10 Sequence Diagram Menu Login Admin 46

Gambar 3.11 Flowmap Proses Pencarian dengan *Levenstein Distance* 47

Gambar 3.12 Tampilan Halaman Utama 49

Gambar 3.13 Tampilan Halaman Menu Tentang Aplikasi 50

Gambar 3.14 Tampilan Menu Login Admin 51

Gambar 3.15 Tampilan Menu Tambah Data Kamus 52

Gambar 3.16 Tampilan Menu Ubah Data Kamus 53

Gambar 3.17 Tampilan Menu Lihat Data 54

Gambar 3.18 Tampilan *Script Edittext* di layout_pencarian.xml 66

UNNES
UNIVERSITAS NEGERI SEMARANG

Gambar 3.19 Tampilan *Script Listview* untuk Tampilan *Autocomplete* 67
Gambar 3.20 Tampilan Script Deklarasi Tampilan 67
Gambar 3.21 Tampilan Script Isi Method setData 68
Gambar 3.22 Tampilan Script onItemClick 68
Gambar 3.23 Tampilan Script Deklarasi Tombol Search 69
Gambar 3.24 Script dari Tombol Pencarian Ketika Diklik 70
Gambar 3.25 Kode untuk Memanggil *Method* dari *Levenshtein* 70
Gambar 3.26 Isi dari *Method* Levenshtein yang Merupakan Algoritma 71
Gambar 4. 1 Tampilan *Script Edittext* di layout_pencarian.xml 66
Gambar 4. 2 Tampilan *Script Listview* untuk Tampilan *Autocomplete* 67
Gambar 4. 3 Tampilan *Script* Deklarasi Tampilan 67
Gambar 4. 4 Tampilan *Script* Isi *Method* setData 68
Gambar 4. 5 Tampilan Script onItemClick 68
Gambar 4. 6 Tampilan *Script* Deklarasi Tombol *Search* 69
Gambar 4. 7 *Script* dari Tombol Pencarian Ketika Diklik 70
Gambar 4. 8 Kode untuk Memanggil *Method* dari *Levenshtein* 70
Gambar 4. 9 Isi dari *Method* Levenshtein yang Merupakan Algoritma *Levenshtein distance* 71
Gambar 4.10 Tampilan dari *Splash Screen* Aplikasi KBBI 72
Gambar 4.11 Tampilan Menu Utama Pencarian Kata 73
Gambar 4.12 Tampilan Proses Pencarian di aplikasi KBBI terdahulu 74
Gambar 4.13 Tampilan Ketika Tidak Terdapat Koneksi Internet 75
Gambar 4.14 Tampilan Menu ketika Tombol Menu Diklik 76
Gambar 4.15 Tampilan Menu Tentang Aplikasi 77
Gambar 4.16 Tampilan Menu Keluar 78
Gambar 4.17 Tampilan Menu Halaman Admin 79
Gambar 4.18 Tampilan Menu Tambah Data Kamus 79
Gambar 4.19 Tampilan Menu Lihat Data 80

Gambar 4.20 Tampilan Menu Detail Lihat Data 81
Gambar 4.21 Tampilan Menu Ubah Data 81
Gambar 4.22 Tampilan Menu Logout 82
Gambar 4.23 Tampilan Menu Lihat Data 83
Gambar 4.24 Tampilan Hasil Pengujian Kata Faseh 92
Gambar 4.25 Tampilan Hasil Pengujian Kata Tasyaruf 94
Gambar 4.26 Tampilan Hasil Pengujian Kata Jebt 95
Gambar 4.27 Tampilan Hasil Pengujian Kata Genecl 97
Gambar 4.28 Tampilan Hasil Pengujian Kata Zibro 98
Gambar 4.29 Tampilan Hasil Pengujian Kata Habtasi 100
Gambar 4.30 Sampel Data untuk *Autocomplete* 104
Gambar 4.31 Nilai Hasil Evaluasi 105
Gambar 4.32 Grafik Perbandingan Algoritma dan Pencarian Tradisional 106

UNNES
UNIVERSITAS NEGERI SEMARANG

DAFTAR TABEL

Tabel 2. 1 Tabel Matriks Perhitungan *Levenshtein distance* 23
Tabel 2. 2 Tabel Matriks Perhitungan *Levenshtein distance* 23
Tabel 2. 3 Tabel Simbol *Flowchart* Beserta Deskripsinya 30
Tabel 3. 1 Pendefinisian Aktor 38
Tabel 3. 2 Identifikasi Use Case 39
Tabel 3. 3 Tabel Pengujian *Autocomplete* 57
Tabel 3. 4 Tabel Pengujian Algoritma *Levenstein Distance* 58
Tabel 3. 5 Panjang Interval 63
Tabel 3. 6 Perhitungan kata dengan algoritma *levenshtein distance* 65
Tabel 4. 1 Pengujian Terhadap Keseluruhan Sistem Aplikasi KBBI 83
Tabel 4. 2 Tabel Pengujian *Autocomplete* 85
Tabel 4. 3 Pengujian Pemberian Saran dengan *Levenshtein distance* 87
Tabel 4. 4 Hasil Pengujian Keberfungsian Sistem Aplikasi KBBI 88
Tabel 4. 5 Hasil Pengujian Fitur *Autocomplete* 89
Tabel 4. 6 Hasil Pengujian Algoritma *Levenshtein distance* 91
Tabel 4. 7 Tabel Matriks Perhitungan *Levenshtein distance* 93
Tabel 4. 8 Jumlah Jarak Perbandingan Antar Kata 93
Tabel 4. 9 Tabel Matriks Perhitungan *Levenshtein distance* 94
Tabel 4.10 Jumlah Jarak Perbandingan Antar Kata 95
Tabel 4.11 Tabel Matriks Perhitungan *Levenshtein distance* 96
Tabel 4.12 Jumlah Jarak Perbandingan Antar Kata 96
Tabel 4.13 Tabel Matriks Perhitungan *Levenshtein distance* 97
Tabel 4.14 Jumlah Jarak Perbandingan Antar Kata 98
Tabel 4.15 Tabel Matriks Perhitungan *Levenshtein distance* 99
Tabel 4.16 Jumlah Jarak Perbandingan Antar Kata 99
Tabel 4.17 Tabel Matriks Perhitungan *Levenshtein distance* 100

UNNES
UNIVERSITAS NEGERI SEMARANG

Tabel 4.18 Jumlah Jarak Perbandingan Antar Kata..... 101
Tabel 4.19 Hasil Pengujian Efektivitas *Autocomplete*..... 101
Tabel 4.20 Hasil Pengujian Efektivitas *Levenshtein distance*..... 102
Tabel 4.21 Saran yang Muncul pada Fitur Autocomplete 104
Tabel 4.22 Hasil Evaluasi Penelitian 105

UNNES
UNIVERSITAS NEGERI SEMARANG

DAFTAR LAMPIRAN

Lampiran 1 Angket Pengujian Aplikasi KBBI 112
Lampiran 2 Surat Kelengkapan 113
Lampiran 3 Surat Kelengkapan SK Penguji 114

UNNES
UNIVERSITAS NEGERI SEMARANG

BAB I

PENDAHULUAN

1.1 Latar Belakang

Kegiatan berbahasa tidak dapat dipisahkan dalam kehidupan umat manusia. Untuk kegiatan sehari-hari, kerja maupun sekolah, penggunaan bahasa sangat dibutuhkan, terlebih lagi dengan berbahasa yang baik dan benar. Dan agar bisa berjalan dengan baik, pengembangan bahasa diperlukan sebagai upaya untuk meningkatkan mutu bahasa dan memenuhi berbagai keperluan dalam kehidupan masyarakat. Beberapa contoh pengembangan bahasa tersebut yaitu melalui penelitian, pembakuan, dan pemasyarakatan (Pusat Bahasa, 2008).

Pembuatan kamus merupakan suatu usaha untuk menyusun atau membukukan bahasa menjadi bagian dari pembakuan bahasa tersebut. Pembukuan ini dilakukan agar bahasa dapat dipelajari terus menerus sampai di masa yang akan datang. Selain itu juga menjadi simpanan tertulis bagi bangsa Indonesia. Kamus besar sendiri memiliki makna kamus yang mencatat kekayaan dalam bentuk lema atau entri, lengkap dengan nuansa dan maknanya. Meningkatnya jumlah lema dan sublema pada kamus besar bahasa Indonesia yang terbaru yaitu KBBI edisi keempat, yakni berjumlah 90.049, yang terdiri atas lema pokok 41.250 dan sublema 48.799, serta pribahasa 2.036, tentunya membutuhkan banyak lembaran dalam pembuatan bukunya (Pusat Bahasa, 2008).

Kamus Besar Bahasa Indonesia yang bersifat daring atau *online* saat ini sudah banyak dibuat ke dalam bentuk perangkat lunak, baik itu dalam bentuk web, aplikasi *desktop*, maupun aplikasi dalam versi *mobile* seperti di *windows phone, iphone*, maupun *android*. Sehingga banyak membantu pengguna dalam memberikan kemudahan untuk mencari informasi kata yang dibutuhkan. Karena sifatnya yang dalam jaringan, jadi kamus ini bisa diakses di mana saja asalkan pengguna terhubung dengan jaringan internet atau *online* baik itu menggunakan *wifi* maupun sambungan data dari *provider*. Untuk penggunaan aplikasi di *desktop* biasanya berbentuk *application web*, sehingga pengguna diarahkan untuk mengunjungi situs dari kamus *online* tersebut seperti diantaranya kbbi.web.id, badanbahasa.kemendikbud.go.id, kamus.ugm.ac.id dan masih banyak lagi. Dengan adanya kamus dalam jaringan ini, pengguna kini tidak perlu disibukkan untuk pergi ke perpustakaan hanya untuk mencari kata yang dibutuhkan dalam kamus. Pengguna dimudahkan dengan sifatnya yang dapat diakses di mana saja dan kapan saja. Bahkan tidak perlu membeli kamus dalam bentuk buku yang harganya bisa dikatakan mahal, hanya perlu mengakses situs web ataupun mengunduh aplikasi dan kemudian dipasang di *smartphone* pengguna.

UNNES
UNIVERSITAS NEGERI SEMARANG

Pada perangkat yang bersistem operasi *android* pengguna hanya perlu memasang aplikasi KBBI yang kini banyak terdapat di Play Store Google baik yaang gratis maupun yang berbayar atau pro. Kamus daring di *android* berarti bahwa basis data atau penyimpanan data kamus disimpan di *server online* seperti MySQL sehingga pembuat aplikasi harus membuat *web hosting* untuk

menyimpan datanya di *server*. Ketika proses pencarian kata dilakukan, akan terjadi proses pencarian ke *server* yang membutuhkan jaringan data.

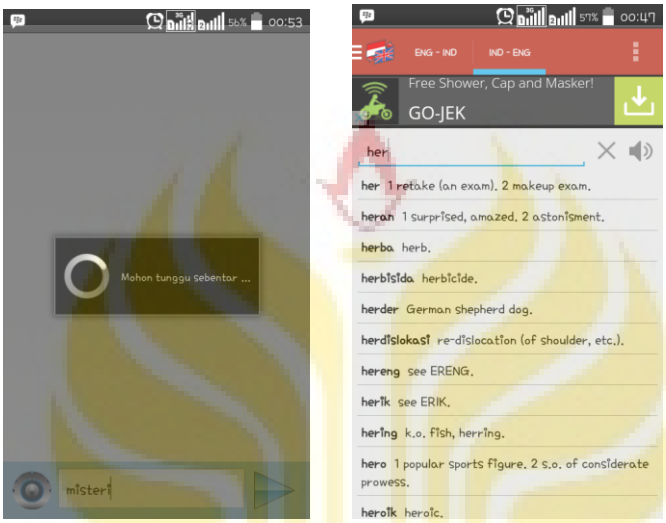
Berbeda halnya dengan kamus yang sifatnya *offline* atau luar jaringan (luring), pengguna tidak perlu menggunakan sambungan internet agar bisa mengakses data dalam kamus tersebut. Hal ini dikarenakan basis datanya terletak dalam aplikasi itu sendiri dan tidak disimpan dalam *server online*. Selain itu untuk proses pencarian katanya bisa dikatakan cepat, karena tidak tergantung pada sambungan data internet. Sehingga hal ini memudahkan bagi pengguna yang ingin menggunakannya dalam waktu singkat dan cepat. Namun aplikasi yang sifatnya luring ini biasanya memiliki ukuran penyimpanan yang lebih besar daripada yang daring, karena aplikasi daring tidak menempatkan basis data pada aplikasi sehingga ukuran lebih kecil. Sedangkan aplikasi luring untuk basis datanya saja bisa mencapai 16 MB dengan jumlah data kurang lebih 20.000 perbendaharaan kata.

Untuk lebih memudahkan pengguna dalam proses pencarian kata di dalam kamus baik itu daring maupun luring, pengembang aplikasi bisa menambahkan fitur *autocomplete* dalam aplikasinya. Fitur *autocomplete* yang berfungsi untuk memberikan pilihan saran kata ketika mengetik ini merupakan fitur yang disediakan baik itu di *browser*, *search engine*, pemrosesan kata, maupun pada baris perintah. Selain itu fitur ini mampu memberikan saran kata yang pengguna inputkan tanpa harus ditulis secara keseluruhan (Yao, 2013).

Fitur *autocomplete* ini sudah tersedia di ADT (Android Developer Tools), namun tidak semua pengembang menerapkan fitur tersebut. Sehingga

UNNES
UNIVERSITAS NEGERI SEMARANG

pembuat aplikasi bisa mengembangkannya untuk diterapkan pada berbagai macam aplikasi. Gambar 1.1 di bawah ini merupakan contoh aplikasi KBBI yang tidak menggunakan fitur *autocomplete* dalam bentuk dalam jaringan dan yang menggunakan *autocomplete* dalam bentuk luar jaringan.



Gambar 1. 1 KBBI daring tanpa fitur *autocomplete* dan KBBI dengan *autocomplete* di luar jaringan.

Aplikasi yang ditunjukkan pada gambar 1.1 di sebelah kanan yaitu aplikasi Kamusku dan gambar kiri yaitu dari aplikasi KBBI, dapat diketahui bahwa pengguna harus menunggu lama untuk proses pencarian data didalam basis data, karena basis data ini disimpan di *server online*. Sehingga kecepatan proses pencarian tergantung dari kecepatan akses internet pada masing-masing *provider* yang terpasang di *smartphone* pengguna. Jika sinyal internet yang digunakan cepat, maka tidak perlu menunggu waktu lama untuk mendapatkan hasil yang diinginkan, namun jika sebaliknya maka hal ini tentu akan sangat memakan waktu yang lama dan membosankan, sedangkan pengguna harus menemukan kata

tersebut dengan cepat dan tepat. Selain basis datanya yang mengharuskan pengguna untuk memiliki koneksi internet, dalam aplikasi KBBI tersebut juga tidak terdapat fitur *autocomplete* dalam proses pencariannya. Sehingga pengguna harus mengetahui keseluruhan kata untuk dimasukkan agar dapat diketahui hasil pencariannya.

Dengan pencarian model seperti ini, pengguna dituntut untuk mengetahui setiap kata yang ingin dicari dalam kamus tersebut. Jika salah menuliskan kata, maka sistem menampilkan bahwa kata yang dicari tidak ditemukan. Padahal hanya terjadi kesalahan 1 huruf, namun tetap tidak dapat dimunculkan kata yang hampir mirip tersebut. Untuk pengguna yang sebelumnya memang tidak mengetahui kata asli dari kata yang ingin dicari (terutama pengguna dari luar Indonesia) tentu akan merasa kesulitan untuk melakukan pencarian.

Dari berbagai macam algoritma pencarian yang ada, seperti algoritma *Knuth-Morris-Pratt* yang mencocokkan kata yang ada dengan *pattern* yang tersedia di dalam basis data (Haryanto, 2011). Dengan algoritma ini proses pencarian akan berlanjut sampai *pattern* dan teks cocok satu sama lain, dan ini memakan waktu yang lama jika teks yang ada jumlahnya banyak. Maupun pencarian dengan metode *sequential search* di mana pencarian ini dilakukan dengan membandingkan setiap elemen larik satu persatu secara beruntun sampai dengan elemen yang dicari ditemukan (Sembiring, 2013). Namun dari kedua metode atau algoritma tersebut tidak dapat menambahkan, mengubah, maupun menghapus karakter. Sedangkan yang dibutuhkan dalam proses pencarian kata di

UNNES
UNIVERSITAS NEGERI SEMARANG

aplikasi KBBI ini adalah algoritma yang dapat melakukan proses-proses tersebut untuk melakukan pengoreksian kata. Untuk itu algoritma *levenshtein distance* dibutuhkan untuk mengubah suatu *string* ke *string* yang lain, di mana operasi-operasi tersebut adalah operasi penyisipan, penghapusan, atau penyubstitusian sebuah karakter (Dwitiyastuti, t.t.).

Untuk memudahkan pengguna dalam melakukan pencarian kata dalam Kamus Besar Bahasa Indonesia berbasis *android* ini, diterapkan fitur di mana ketika pengguna mengetikkan kata yang ingin dicari di kolom pencarian. Lalu akan ada pilihan saran kata yang mendekati dengan kata yang pengguna masukkan. Fitur ini disebut juga dengan *autocomplete*. Fitur ini bekerja ketika pengguna mengetikkan huruf pertama dari suatu kata. Jika kata yang diinginkan belum diketemukan, maka pengguna bisa melanjutkan untuk memasukkan huruf kedua, maupun ketiga, dan seterusnya. Dan ketika kata yang dimasukkan salah atau tidak ada, maka akan dilanjutkan dengan proses berjalannya algoritma *levenshtein distance* untuk memberikan sugesti atau saran kata yang mendekati dengan kata yang dimasukkan tersebut.

Dengan penelitian yang berjudul "Implementasi Fitur *Autocomplete* dan Algoritma *Levenshtein Distance* untuk Meningkatkan Efektivitas Pencarian Kata di Kamus Besar Bahasa Indonesia (KBBI)", diharapkan dapat membantu pengguna untuk memudahkan proses pencarian kata di Kamus Besar Bahasa Indonesia ini.

UNNES
UNIVERSITAS NEGERI SEMARANG

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka diperoleh rumusan masalah sebagai berikut :

1. Bagaimana mengimplementasikan fitur *autocomplete* dan algoritma *levenshtein distance* dalam aplikasi Kamus Besar Bahasa Indonesia (KBBI)?
2. Apakah implementasi fitur *autocomplete* dan algoritma *levenshtein distance* efektif digunakan pada pencarian kata dalam aplikasi KBBI?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah dikemukakan di atas, tujuan dari penelitian ini yaitu :

1. Untuk mengetahui bagaimana cara mengimplementasikan fitur *autocomplete* dan algoritma *levenshtein distance* pada aplikasi Kamus Besar Bahasa Indonesia (KBBI).
2. Untuk mengetahui efektifivitas penggunaan fitur *autocomplete* dan algoritma *levenshtein distance* dalam proses pencarian kata di KBBI.

UNNES
UNIVERSITAS NEGERI SEMARANG

1.4 Manfaat Penelitian

Hasil penelitian ini diharapkan :

1. Memberikan kemudahan kepada user dengan menambahkan fitur *autocomplete* dan pengoreksian kata dengan algoritma *levenshtein distance* pada pencarian di aplikasi Kamus Besar Bahasa Indonesia.

2. Mengoptimalkan fungsi dari aplikasi Kamus Besar Bahasa Indonesia yang telah ada sebelumnya.
3. Memberikan kemudahan kepada user dalam pencarian kata di Kamus Besar Bahasa Indonesia.

1.5 Batasan Masalah

Penelitian skripsi ini dibatasi oleh permasalahan sebagai berikut :

1. Pembuatan Kamus Besar Bahasa Indonesia ini berbasis *android*.
2. Untuk meningkatkan keefektifan pencarian kata dalam kamus digunakan fitur *autocomplete* dan algoritma *levenshtein distance* untuk pengecekan kesalahan kata.
3. Digunakan pengujian *black box* untuk menguji hasil akhir dari aplikasi KBBI ini.
4. Aplikasi KBBI ini bersifat *offline* dengan basis data menggunakan SQLite.
5. Hanya dapat diakses dan dipasang pada perangkat bersistem operasi android versi 3.2 keatas yaitu Ice Cream Sandwich, Jellybean, Kitkat, dan Lollipop.
6. Aplikasi ini dibuat untuk Kamus Besar Bahasa Indonesia (KBBI).
7. Aplikasi KBBI ini hanya bersifat prototipe, dan basis datanya terbatas (tidak keseluruhan).

UNNES
UNIVERSITAS NEGERI SEMARANG

8. Jenis *autocomplete* yang digunakan dalam aplikasi ini adalah *autocomplete* yang mendeteksi huruf pertama dan yang selanjutnya, pada masing-masing kata dalam suatu frasa.

1.6 Definisi Operasional Judul Penelitian

Dari judul penelitian ini dapat didefinisikan bahwa :

1. Fitur Autocomplete

Autocomplete merupakan suatu fitur dalam kotak pencarian, dimana ketika pengguna mengetikkan kata yang ingin dicari maka akan muncul pilihan kata yang mirip dengan kata yang dimasukkan. *Autocomplete* melibatkan program yang dapat melakukan prediksi terhadap sebuah kata atau frasa yang pengguna ingin tulis tanpa harus menulis keseluruhan kata atau frasa secara lengkap (Banowosari, 2014).

2. Algoritma Levenshtein Distance

Algoritma *levenshtein distance* merupakan matrik yang digunakan untuk mengukur perbedaan jarak antara dua *string* atau kata. 3 operasi utama yang dilakukan oleh algoritma *levenshtein distance* ini yaitu penyisipan karakter, penghapusan karakter, dan pengubahan atau penukaran karakter (Dewantara, t.t)

3. Efektivitas Pencarian Kata

Efektivitas pencarian kata merupakan suatu bagian untuk mengetahui apakah pencarian kata yang menggunakan fitur *autocomplete* dan algoritma *levenshtein distance* ini efektif untuk digunakan pada aplikasi KBBI atau

tidak. Proses ini dilakukan dengan menguji fitur dan algoritma tersebut kepada beberapa responden. Responden akan diminta untuk mengisi angket yang berisi beberapa pernyataan, di mana ada dua hal utama yang diujikan yaitu mengenai kesesuaian kata yang keluar dengan kata yang dimasukkan dan kecepatan waktu munculnya saran.

UNNES
UNIVERSITAS NEGERI SEMARANG

BAB II

LANDASAN TEORI

2.1 Kamus

Kamus merupakan sekumpulan informasi tentang sebuah kata atau kombinasi kata. Sekumpulan informasi yang berhubungan dengan sebuah kata atau suatu kombinasi kata ini disebut juga dengan aran (entry). Kata yang diterangkan ini tertulis di awal aran dan disebut kata aran. Jadi, kamus adalah sekumpulan aran (Suryawinata, 2003).

Terdapat beberapa macam kamus menurut Suryawinata (2003:28) yaitu menurut bahasa yang digunakan, kamus dapat dibedakan menjadi kamus ekabahasa, kamus dwibahasa, dan kamus nekabahasa. Kamus ekabahasa merupakan kamus yang hanya menggunakan satu bahasa saja, contohnya adalah Kamus Besar Bahasa Indonesia, *Oxford English Dictionary* dan sebagainya. Kamus dwibahasa merupakan kamus yang menggunakan dua bahasa, contohnya seperti kamus Indonesia-Inggris, oleh John M. Echols dan Hassan Shadily maupun kamus Indonesia-Madura yang diterbitkan oleh Balai Bahasa Provinsi Jawa Timur. Sedangkan kamus nekabahasa berisi keterangan tentang kata aran didalam dua bahasa atau lebih. Contohnya adalah kamus bahasa Inggris-Indonesia-Arab.

UNNES
UNIVERSITAS NEGERI SEMARANG

Untuk fungsinya sendiri, menurut Bapak Leksikologi Inggris yaitu Samuel Johnson yang menyusun *Dictionary of the English Language* (1755), menyatakan bahwa kamus berfungsi untuk memelihara kemurnian bahasa (Pusat Bahasa, 2008). Karena semakin dengan berkembangnya zaman, bahasa juga turut berkembang. Dan seperti yang kita ketahui bahwa dalam masyarakat Indonesia sendiri dikenal terdapat bahasa-bahasa gaul yang digunakan dalam kehidupan sehari-hari. Dengan adanya kamus ini Bahasa Indonesia akan tetap terjaga kemurniannya, meskipun telah ada bahasa-bahasa baru yang muncul di masyarakat. Sehingga di masa depan masyarakat yang telah berkembang tetap memiliki pegangan bahasa yang telah disusun sejak lama .

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 membahasa 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.

UNNES
UNIVERSITAS NEGERI SEMARANG

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 didalam 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.

UNNES
UNIVERSITAS NEGERI SEMARANG

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 kamus pun 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 yaitu *hukum* dan memiliki entri atau lema *penghukum*, *penghukuman*, *hukuman*, dan *kehukuman* (Wiyanto, 2005).

2.2 KBBI

Kamus besar merupakan kamus yang mencatat kekayaan suatu bahasa tertentu, yang disusun dalam bentuk lema atau entri, dan disertai pula dengan maknanya. Makna kata yang diuraikan di kamus itu tertuang dalam bentuk definisi, deskripsi, contoh, sinonim (suatu kata atau bahasa yang memiliki makna sama), ataupun dengan parafrasa (pengungkapan kembali suatu kata). Uraian tersebut juga disertai dengan label pemakaian suatu kata dan maknanya; daerah atau kelompok sosial yang menggunakan kata tersebut dan disertai maknanya

UNNES
UNIVERSITAS NEGERI SEMARANG

(label dialek regional atau dialek sosial), atau yang masih digunakan (label dialek temporal); dan dengan etimologi yang menjelaskan perkembangan bentuk dan makna kata sejak permulaan kata itu dipakai dalam bahasa, baik kata asli maupun kata serapan bahasa lain (Pusat Bahasa, 2008).

KBBI terbit pertama kali pada tahun 1988, kemudian terbit edisi kedua tahun 1991, dan edisi ketiga pada tahun 2001. Kini umur KBBI telah 20 tahun hadir dimasyarakat Indonesia, dari edisi pertama hingga edisi ketiga terdapat perkembangan yang signifikan, terutama dalam hal kosakata, baik umum maupun khusus. Pada edisi pertama jumlah lema yang dimuat sekitar 62.000, edisi kedua sekitar 72.000, dan edisi ketiga sekitar 78.000 dan peribahasa sebanyak 2.034 (Pusat Bahasa, 2008).

Kamus Besar Bahasa Indonesia yang diterbitkan oleh pusat bahasa pada edisi keempat mengalami banyak peningkatan jumlah lema dan sublema, yakni 90.049, yang terdiri atas lema pokok 41.250 dan sublema 48.799, serta peribahasa sebanyak 2.036, namun pada edisi ke empat ini hanya menambahkan 2 peribahasa karena bentuk bahasa yang tidak berkembang, tidak seperti kata atau lema yang terus mengalami perkembangan atau penambahan (Pusat Bahasa, 2008). Banyaknya lema yang ada pada KBBI membuat pencarian pada aplikasi menjadi lebih lama jika proses pencarian dilakukan secara satu persatu sesuai *index* yang ada. Untuk memudahkan proses pencarian tersebut, dibutuhkan suatu fitur yang dapat membantu pengguna dalam proses pencarian, yaitu dengan menggunakan fitur *autocomplete*, yang akan menyaring pencarian sesuai dengan huruf awal yang dimasukkan pengguna.

UNNES
UNIVERSITAS NEGERI SEMARANG

Dalam KBBI edisi keempat ini, terdapat bagian catatan perbendaharaan kata dalam bahasa Indonesia yang meliputi : kata-kata umum hasil inventarisasi selama beberapa tahun terakhir ini; lalu kata-kata yang termuat dalam kamus-kamus lain, yang sudah melalui seleksi; kata-kata dari pelbagai daerah, khususnya kosakata yang berkaitan dengan budaya-budaya di Indonesia yang telah diteliti penggunaannya dalam kehidupan sehari-hari; dan istilah dari berbagai bidang kehidupan dan cabang ilmu pengetahuan yang menurut pakar yang bersangkutan pantas dimuat dalam Kamus Besar Bahasa Indonesia (Pusat Bahasa, 2008).

Penggunaan KBBI yang telah meluas ke segala kalangan baik itu sastrawan, penulis, mahasiswa maupun pekerja lain membuat KBBI ini terus mendapatkan perbaikan dari waktu ke waktu. Sehingga kini KBBI sudah ada sampai edisi ke empat sejak pertama kali disusun dan diterjunkan kemasyarakat.

2.3 KBBI Digital

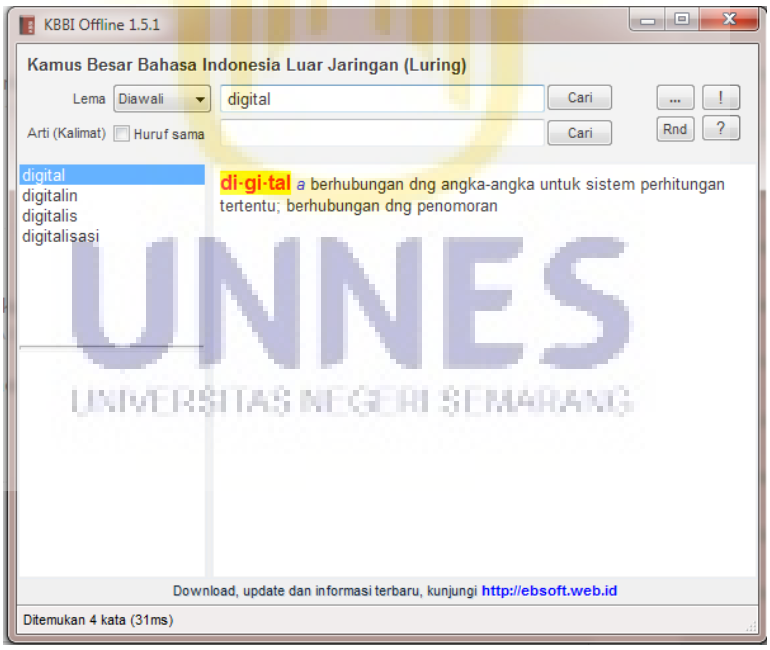
KBBI digital merupakan bentuk lain dari KBBI yang berupa cetakan atau dalam bentuk buku. Dikatakan bentuk digital karena memang dibuat untuk dapat digunakan pada perangkat elektronik seperti komputer, tablet, maupun *handphone*. Dewasa ini sudah begitu banyak KBBI digital yang muncul dalam bentuk elektronik. Cara ini digunakan untuk memudahkan pengguna mengakses KBBI.

Beberapa macam KBBI digital yang dapat ditemui dan diakses oleh pengguna terutama *web* adalah *kbbi.web.id*, *badanbahasa.kemdikbud.go.id*, *kamusbahasaindonesia.org*, *kamus-kbbi.com*, *kamuscektkp.com*, dan masih ada

UNNES
UNIVERSITAS NEGERI SEMARANG

beberapa macam kamus lain lagi yang terdapat di internet dan dapat diakses secara gratis oleh siapapun asalkan terhubung dengan koneksi internet, seperti pada gambar 2.1. Berikut ini merupakan contoh *screenshot* dari tampilan KBBI digital dalam bentuk *web* dan aplikasi *offline* pada gambar 2. 2 :

Gambar 2.1 Contoh KBBI Digital Berupa di badanbahasa.kemdikbud.go.id



Gambar 2.2 Contoh KBBI Digital dalam Bentuk Luar Jaringan

Untuk di *smartphone*, maka pengguna bisa langsung mengunduh aplikasi tersebut di *play store* dengan mengetikkan KBBI di kotak pencarian, dan namanya pun hampir sama semua yaitu menggunakan nama KBBI, namun tentu setiap aplikasi memiliki keistimewaan maupun kekurangan masing-masing. Gambar 2.3 di bawah ini merupakan contoh *screenshot* tampilan dari aplikasi di *smartphone* android :

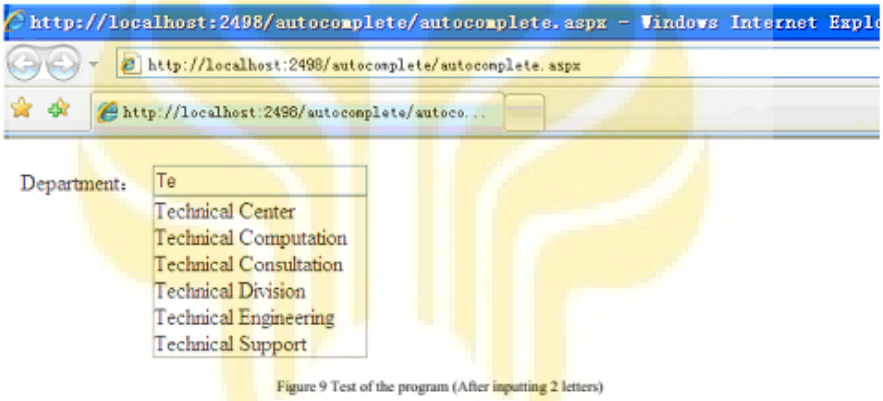
Gambar 2.3 Contoh *Screenshot* Aplikasi KBBI Berbasis Android

Dari tiga contoh aplikasi KBBI di atas, kesemuanya tidak menggunakan fitur *autocomplete*. Sehingga pengguna harus mengetikkan kata secara keseluruhan untuk menyelesaikan proses pencarian.

2.4 Fitur *Autocomplete*

Autocomplete merupakan suatu fitur dalam kotak pencarian, di mana ketika pengguna mengetikkan kata yang ingin dicari maka akan muncul pilihan kata yang mirip dengan kata yang dimasukkan. *Autocomplete* melibatkan program

yang dapat melakukan prediksi terhadap sebuah kata atau frasa yang pengguna ingin tulis tanpa harus menulis keseluruhan kata atau frasa secara lengkap (Banowosari, 2014). Keuntungan penggunaan *autocomplete* ini adalah komputer membantu pengguna untuk mengingat kata yang akan dimasukkan. Selain itu kesalahan pada proses memasukkan kata dapat diminimalisir. Dan keuntungan lainnya yaitu mempercepat proses interaksi pencarian kata (Hyvonen, 2006). Gambar 2.4 di bawah ini merupakan contoh *autocomplete* :



Gambar 2.4 Contoh dari Proses *Autocomplete* (Yao, 2013)

Fitur *autocomplete* pada sistem operasi android bekerja ketika penulis menuliskan kata pada kotak pencarian. Program yang melakukan prediksi akan mencari satu atau lebih kemungkinan kata sebagai pilihan. Jika kata yang dimaksud ada dalam pilihan tersebut, maka pengguna bisa langsung memilih pilihan tersebut. Dan jika kata yang dimaksud tidak ada dalam pilihan prediksi maka penulis harus menulis huruf selanjutnya (Kusuma, 2012).

Ketika mengetikkan suatu kata dalam kotak pencarian, agar kata yang muncul bisa cepat terdeteksi maka diperlukan kata dasar yang tepat dan benar

(Hyvonen, 2006). Fitur *autocomplete* ini pun sudah banyak digunakan diberbagai macam lingkup, seperti : mesin pencari, kamus digital, *browser*, bahkan dalam proses penulisan di aplikasi pemrograman seperti *dreamweaver*, *eclipse*, maupun yang lainnya sudah menerapkan fitur *autocomplete*. Hal ini tentu membantu bagi para *developer* pemula yang masih belum hapal dengan benar setiap sintak atau kode yang ada pada bahasa pemrograman tersebut.

Autocomplete sendiri ada beberapa jenis, yaitu : (1) *autocomplete suggestion* oleh Google. *Autocomplete* jenis ini digunakan oleh Google pada mesin pencariannya, di mana *autocomplete* ini akan menampilkan perkiraan kata atau frasa yang akan dimasukkan tanpa harus mengetikkan keseluruhan kata (Banowosari, 2014). *Autocomplete* ini melakukan pencarian sesuai dengan apa yang pengguna masukkan pertama kali, dan saran akan muncul berdasarkan kata ataupun huruf yang belum selesai diketik di kotak pencarian. (2) *autocomplete* pada perangkat lunak. *Autocomplete* jenis ini digunakan pada berbagai macam perangkat lunak, seperti Microsoft Excel, Dreamweaver, maupun Eclipse yang akan muncul ketika pengguna memasukkan huruf pertama kali. Pada jenis ini *autocomplete* akan memberikan saran berupa argumen fungsi, dan dengan fitur ini akan memudahkan juga mengoreksi argumen atau kata yang dimasukkan tanpa mengingat formatnya (Dixon, 2007). (3) *autocomplete* pada android. *Autocomplete* ini menunjukkan saran secara otomatis ketika pengguna sedang mengetik, yang di mana pilihan saran ini muncul pada menu *drop down* dan pengguna dapat memilih saran yang sesuai dengan yang dimasukkan (Anonim, t.t). *Autocomplete* ini bekerja ketika pengguna mengetikkan huruf pertama kali,

UNNES
UNIVERSITAS NEGERI SEMARANG

dan akan mendeteksi huruf pertama, pada masing-masing kata dalam suatu frasa. Dan dalam pembuatan aplikasi KBBI ini digunakan *autocomplete* yang sudah tersedia di android.

2.5 Algoritma Levenshtein Distance

Algoritma *levenshtein distance* atau bisa disebut juga dengan *edit distance*, diciptakan oleh Vladimir Levenshtein pada tahun 1965. Perhitungan *edit distance* merupakan suatu perhitungan dari matriks yang digunakan untuk menghitung jumlah perbedaan antara dua *string*. Perhitungan *distance* atau jarak antara kedua *string* ini ditentukan oleh jumlah minimum operasi perubahan untuk merubah dari *string* A menjadi *string* B, yang dihitung dengan menggunakan tabel perhitungan *levenshtein distance*, di mana nilai terakhir yang berada di paling pojok kanan bawah merupakan nilai akhir dari jarak kedua *string*. Yang dimaksud dengan *distance* atau jarak adalah jumlah perubahan yang diperlukan untuk mengubah suatu bentuk *string* ke *string* lainnya. Contohnya yaitu *string* "bentuk" dan "bentik" memiliki jarak 1 karena diperlukan satu operasi untuk mengubah *string* "bentik" menjadi "bentuk" (Dwitiyastuti, 2010). Dalam pengimplementasiaannya, *levenshtein distance* juga digunakan dalam aplikasi *spell checker* (pengkoreksian ejaan), kamus, pengenalan wicara, *search engine* seperti yang dimiliki oleh *Google*, pengoreksian kata kunci, SMS *filtering* maupun pendeteksi plagiat.

Terdapat 3 macam operasi utama yang dapat dilakukan oleh algoritma ini, yaitu :

UNNES
UNIVERSITAS NEGERI SEMARANG

a. Operasi pengubahan karakter

Operasi pengubahan karakter merupakan operasi untuk menukar suatu karakter dengan karakter lain. Seperti contohnya yaitu pengubahan karakter oleh *string* “jija” menjadi jika”, di mana karakter j diubah menjadi karakter k.

b. Operasi penambahan karakter

Operasi penambahan karakter merupakan suatu operasi yang menambahkan karakter tertentu ke dalam suatu *string*. Misalnya yaitu penambahan karakter “a” ke dalam *string* “lemri”, sehingga dengan penambahan ini yang sebelumnya “lemri” menjadi *string* “lemari”. Penambahan karakter pun tidak hanya dilakukan diakhir kata, namun bisa ditambahkan diawal maupun disisipkan ditengah *string*.

c. Operasi penghapusan karakter

Operasi penghapusan karakter merupapkan operasi untuk menghilangkan karakter tertentu pada suatu *string*. Misalkan saja pada kata “jumblah” akan dihapus satu karakter yaitu karakter “b” sehingga menjadi “jumlah”.

UNNES
UNIVERSITAS NEGERI SEMARANG

Misalkan ada contoh jarak dari *string* “kipas” dan “kyps”, berarti memiliki *distance* 3, di mana penjelasan operasi algoritmanya adalah : 1) menghapus huruf ‘y’ pada “kyps”→ “kps”; 2) menambahkan huruf ‘i’ pada “kps”→ “kips”; 3) dan menambahkan huruf ‘a’ pada “kips”→ “kipas”. Dalam proses perubahan tersebut, terdapat dua macam proses, yaitu penambahan huruf a dan i, dan juga penghapusan huruf y,. Algoritma ini berjalan mulai dari pojok kiri

atas sebuah *array* dua dimensi yang telah diisi sejumlah karakter *string* awal dan *string* target dan diberikan nilai *cost*. Nilai *cost* pada ujung kanan bawah di tabel 2.1 menjadi nilai *edit distance* yang menggambarkan jumlah perbedaan dua *string*. Berikut ini merupakan contoh dari perhitungan dengan tabel *levenshtein distance* yaitu dengan membandingkan kata “kipas” dengan “kyps”.

Tabel 2.1 Tabel Matriks Perhitungan *Levenshtein distance*

		K	I	P	A	S
	0	1	2	3	4	5
K	1	0	1	2	3	4
Y	2	1	2	3	4	5
P	3	2	1	0	1	2
S	4	3	2	1	2	3

Contoh lain dari perhitungan *levenshtein distance* dengan menggunakan tabel pada *string* “kamu” dan “kmu”, dapat dilihat pada tabel 2.2 di bawah ini :

Tabel 2. 2 Tabel Matriks Perhitungan *Levenshtein distance*

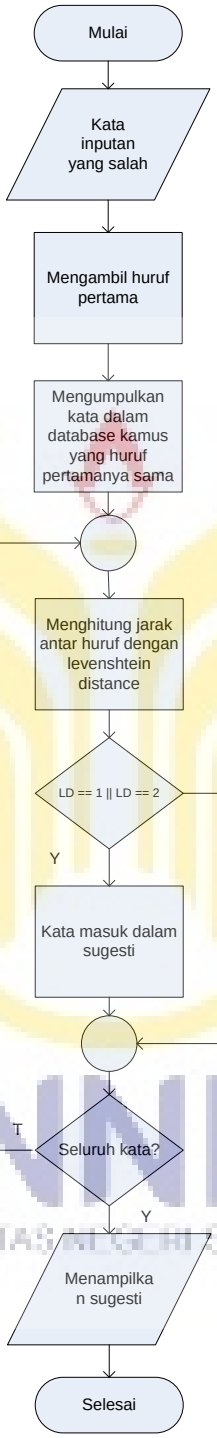
		k	a	m	u
	0	1	2	3	4
k	1	0	1	2	3
m	2	1	2	1	2
u	3	2	1	2	1

Contoh dari perhitungan *levenshtein distance* menggunakan 2 *string* yang berbeda kemudian dihitung *edit distance*-nya pada tabel di atas. Dapat dilihat hasil perhitungan *levenshtein distance* antara 2 *string* ‘kmu’ dan ‘kamu’

adalah 1. Pengecekkan dimulai dari penghitungan awal dari kedua *string* kemudian dilakukan operasi penambahan. Nilai l*evenshtein distance* nya yaitu pada ujung kanan bawah matriks. Hanya ada satu proses penyisipan yang dilakukan yaitu penyisipan karakter 'a' pada *string* 'kmu' sehingga menjadi 'kamu'. Pada pengecekan kata ini, proses perhitungan yang dilakukan pada sejumlah kata yang ada pada basis data SQLite.

Proses berjalannya algoritma *levenshtein distance* ini dimulai dari adanya kata salah yang dimasukkan oleh pengguna, kemudian diambil karakter pertama dari masukkan di kotak pencarian. Setelah itu kata di dalam basis data yang karakter atau huruf pertamanya sama dikumpulkan, untuk kemudian dihitung jaraknya. Lalu dibandingkan apakah LD 1 (masukan kata oleh pengguna) dan LD 2 (kata dalam basis data) sama atau tidak? Jika sama, maka kata tersebut akan masuk kedalam saran yang akan dimunculkan. Namun jika berbeda maka langkah tersebut akan menuju langkah selanjutnya yaitu apakah yang muncul seluruh kata atau tidak? Jika tidak maka akan kembali lagi kelangkah awal yaitu mencari jarak dengan algoritma levenshtein distance. Namun jika yang muncul adalah keseluruhan kata maka akan ditampilkan saran atau prediksi, disini proses berhenti. Lebih singkatnya ditunjukkan pada gambar 2.5 di bawah ini dalam bentuk *flowchart* :

UNNES
UNIVERSITAS NEGERI SEMARANG



Gambar 2.5 *Flowchart* dari Algoritma *Levenshtein Distance* (Dwitiyastuti, t.t)

Berikut ini merupakan pseudocode dari algoritma *levenshtein distance* menurut (Ilmy, 2006) pada gambar 2.6 :

```
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}
ifs1[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]
```

Gambar 2. 6 Pseudocode dari Algoritma *Levenshtein distance*

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

$$Sim = 1 - \frac{d}{max of\ 2\ strings}$$

Keterangan :

Sim = *similarity* / kemiripan

d = nilai *edit distance*

max of 2 string = jumlah maksimal karakter dari kedua kata

2.6 Android

Android merupakan suatu sistem operasi yang berbasis *linux* atau bersifat *open source* (terbuka), di mana penggunanya bisa mengembangkan aplikasinya sendiri sesuai yang diinginkan dan dibutuhkan. Android kini sudah digunakan pada berbagai macam peralatan elektronik seperti *tablet* maupun di *handphone*. Beberapa diantara merk *handphone* yang menggunakan sistem operasi android adalah Samsung, Motorola, Sony, LG dan lain sebagainya. Android banyak dipilih oleh pengguna karena nyaman dipakai, terdapat berbagai macam aplikasi yang tersedia di *playstore* baik yang gratis maupun yang berbayar, harganya yang terjangkau bagi setiap kalangan, dan yang terpenting adalah pengguna bisa menciptakan aplikasinya sendiri (Bhawiyuga, 2011).

Selain keuntungan tersebut, android yang kini diakuisisi oleh *Google*, dulunya merupakan perusahaan yang berdiri sendiri dengan nama Android inc. (Bhawiyuga, 2011). Dari tahun 2007 hingga akhir tahun 2014 ini, android telah berkembang begitu pesat. Berbagai macam versi android pun telah dikembangkan untuk meningkatkan fungsi android sebagai sistem operasi yang kini banyak dilirik oleh berbagai macam produsen *smartphone* maupun *tablet*. Dari tahun 2007 ini, android pertama yang diluncurkan dinamakan Cupcake, Donut, Éclair,

Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, JellyBean, Kitkat, dan Lollipop (Dito, 2014).

Dari proses berkembangnya versi android dari tahun 2007 hingga sekarang, menurut Supriyanto (2012) secara umum android mendukung beberapa fitur-fitur yang sudah terkonfigurasi. Fitur-fitur tersebut diantaranya : adanya fasilitas penyimpanan, di mana penyimpanan ini menggunakan SQLite yang merupakan suatu *embedded* basis data; lalu didukungnya koneksi GSM/EDGE, CDMA, EV-DO, dan lain sebagainya; pada sisi *hardware* terdapat fitur *accelerometer* sensor, kamera, digital *compass*, *proximity* sensor, dan GPS; mendukung layar yang *multi-touch* (dapat mengenali lebih dari satu sentuhan) dan *multi-tasking* (dapat melakukan lebih dari satu membuka aplikasi atau melakukan kegiatan); adanya *flash* pada kamera yang dimulai dari android versi 2.3; adanya dukungan SMS dan MMS, *bluetooth*, media dan *web browser* yang menggunakan *open-source WebKit.*

2.7 Perangkat Perancangan Sistem

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

UNNES
UNIVERSITAS NEGERI SEMARANG

1. Flowchart

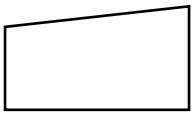
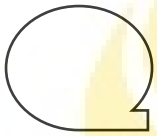
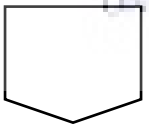
Flowchart merupakan suatu kumpulan atau aliran bagan yang menggambarkan terjadinya proses prosedur atau program. Jadi dalam suatu *flowchart* yang dibuat, bisa menggambarkan bagaimana suatu program itu berjalan. Sehingga jika penjelasan secara deskripsi masih kurang dipahami oleh orang lain, maka ada 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. 3 Tabel Simbol *Flowchart* Beserta Deskripsinya

Simbol	**Definisi**
Simbol arus/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/penghubung	Menyatakan sambungan dari proses satu ke proses lainnya namun dalam halaman yang sama.
Decision/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*.

Simbol	Definisi
/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 :

- 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.

- 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.

- Package diagram

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

- Activity diagram

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

- Deployment diagram

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

UNNES
UNIVERSITAS NEGERI SEMARANG

- Statechart diagram

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

- Collaboration diagram

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

- Sequence diagram

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

2.9 Penelitian yang Relevan

Berdasarkan penelitian yang dilakukan oleh Yoke Yie Chen, Suet-Peng Yong, dan Adzlan 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 Riya Mary Abraham, Sayali Nishikant C, Jaya Subalakshmi R Dan Ch.S.N Iyengar, menyimpulkan bahwa dengan menggunakan algoritma *levenshtein distance* pada

UNNES
UNIVERSITAS NEGERI SEMARANG

pencarian di *cloud* dapat memberikan pencarian yang lebih banyak dibandingkan dengan menggunakan pencarian tradisional.

Lalu penelitian yang dilakukan oleh Zhiqiang Yao dan Dr. Abhijit Ph.D 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 Aouragh Si Lhoussain, Gueddah Hicham, dan Yousfi Abdellah menyimpulkan bahwa dengan mengkombinasikan model *bi-grams* dengan *levenshtein distance* dapat meningkatkan kepuasan dalam solusi metode tersebut..

UNNES
UNIVERSITAS NEGERI SEMARANG

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil pengembangan aplikasi yang telah dilakukan pada aplikasi KBBI ini dapat ditarik kesimpulan :

1. Fitur *autocomplete* diimplementasikan pada aplikasi KBBI dengan menambahkan fitur tersebut pada kotak pencarian *edittext* di *eclipse*. Lalu algoritma *levenshtein distance* diimplementasikan pada bagian tombol pencarian yaitu tombol *search* agar memudahkan pengguna ketika terjadi kesalahan pengetikan.
2. Fitur *autocomplete* yang diimplementasikan pada aplikasi KBBI sangat efektif untuk digunakan dengan nilai persentase keefektifan adalah 84.615%. Algoritma *levenshtein distance* yang diimplementasikan pada aplikasi KBBI efektif untuk digunakan dengan nilai persentase keefektifan adalah 76.04 %.

UNNES
UNIVERSITAS NEGERI SEMARANG

5.2 Saran

Saran untuk pengembangan aplikasi KBBI ini adalah :

1. Masih perlu penambahan basis data Bahasa Indonesia.
2. Penambahan menu pencarian kata dan ungkapan daerah, kata dan ungkapan asing, serta sinonim dan akronim agar kamus digital ini lebih lengkap seperti versi cetaknya.

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.

Adriyani, Ni Made Muni., dkk. 2010. *Implementasi Algoritma Levenshtein distance dan Metode Empiris untuk Menampilkan Saran Perbaikan Kesalahaan Pengetikan Dokumen Berbahasa Indonesia*. Universitas Udayana.

Anonim. Tanpa Tahun. *AutocompleteTextView*. http://developer.android.com/. Diakses 19 Oktober 2015.

Banowosari, Lintang Y., dkk. 2014. *Analisis pada Fitur Autocomplete Suggestion dan Semantic pada Pencarian di Mesin Pencari Google*. Prosiding Seminar Ilmiah Nasional Komputer dan Sistem Intelijen (KOMMIT 2014) Vol. 8 Oktober 2014. Diakses 6 Maret 2014.

Basir, Amat. 2014. *Rancang Bangun Sistem Informasi Disposisi Surat Masuk Dinas Pendidikan Kota Semarang*. Skripsi. Semarang: Fakultas Teknik, Universitas Negeri Semarang.

Bhawiyuga, Adhitya. 2011. *Sistem Pelaporan dan Informasi Posisi Kereta Api Berbasis Global Positioning System (GPS) Pada Device Berbasis Android*. Skripsi. Surabaya : Institut Teknologi Surabaya.

Binanto, Iwan. 2014. *Analisa Metode Classic Life Cycle (Waterfall) Untuk Pengembangan Perangkat Lunak Multimedia*. Seminar Nasional Sains dan Teknologi Informasi (SeNASTI) 2014 : 3.

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

Dito. 2014. *Macam Android dari Dulu Hingga Sekarang*. http://dito.blog.uns.ac.id/. Diakses 29 April 2015.

Dixon, Helen. 2007. *Excel 2007 : Beyond the Manual*. Springer-Verlag : New York.

Dwitiyastuti, Rachmania Nur dkk. Tanpa Tahun. *Pengoreksi Kesalahan Ejaan Bahasa Indonesia Menggunakan Metode Levenshtein distance*. http://download.portalgaruda.org/. Diakses 15 Februari 2015.

Fatta, Hanif Al. 2007. *Analisis dan Perancangan Sistem Informasi untuk Keunggulan Bersaing Perusahaan dan Organisasi Modern*. ANDI : Yogyakarta.

Geisler, Reiner. 2015. *Industrial Software Applications*. http://dnb.dnb.de. Diakses 23 Oktober 2015.

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

Haryanto, Edy Victor. 2011. *Rancang Bangun Prototype Mesin Pencari String Menggunakan Metode Fuzzy String Matching*. Konferensi Nasional Sistem dan Informatika, Bali.

Hyvonen, Eero., Eetu Makela. 2006. *Semantic Autocompletion*. Semantic Computing Research Group (SeCo). http://www.seco.tkk.fi/. Diakses 10 Februari 2015.

Ida, Winarti. 2010. *Pengaruh Area Hotspot (Wi-Fi) Bagi Pemenuhan Kebutuhan Informasi Pemustaka di Kantor Perpustakaan Daerah Kabupaten Jepara*. Undergraduate thesis, Faculty of Humanities.

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

Kusuma, Muhammad Wachid. 2011. *Pencocokan String dalam Fitur Autocompletion pada Text Editor atau Integrated Development Environment (IDE)*. http://informatika.stei.itb.ac.id/. Diakses 8 Februari 2015.

Lhoussain, Aouragh Si. 2015. *Adaptating The Levenshtein Distance To Contextual Spelling Correction*. International Journal Of Computer Science And Application. (12) 1 : 127-133.

Malik, Abdul, S.Pd. 2013. *Teknik Pengumpulan Data Angket atau Kuesioner (Data Collection Techniques Or Questionnaire Questionnaire)*. http://pokoe-mimpiku.blogspot.com. Diakses 2 Juni 2015.

Oei, Istijanto. 2005. *Riset : Sumber Daya Manusia*. Gramedia : Jakarta.

Pressman, Roger S, Ph.D. 1997. *Rekayasa Perangkat Lunak*. ANDI and McGraw-Hil Book : Yogyakarta.

Pusat Bahasa. 2008. *Kamus Besar Bahasa Indonesia Pusat Bahasa*. Gramedia Pustaka Utama : Jakarta.

Rahardja, Yani,.dkk. 2008. *Analisis dan Perancangan Mobile-Banking dengan Menggunakan UML*. Jurnal Teknologi Informasi-Aiti. (5) 2 : 101-200.

Sembiring, Jhoni Pranata. 2013. *Perancangan Aplikasi Kamus Bahasa Indonesia-Karo Online Berbasis Web dengan Metode Sequential Search*. Pelita Informatika Budi Darma. Volume 4, No.2.

Supriyanto, Dodit., Rini Agustina. 2012. *Pemrograman Aplikasi Android : Step by Step Membuat Aplikasi Android untuk Smartphone dan Tablet*. Mediakom: Yogyakarta.

Suryawinata, Z & S.Hariyanto. 2003. *Translation: Bahasa dan Penuntun Praktis Menerjemahkan*. Yogyakarta: Kanisius.

Wicaksono, Yogi. 2008. *Membangun Bisnis Online Dengan Mambo*. Elex Media Komputindo: Jakarta.

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

Yao, Zhiqiang. 2013. *Implementation Of The Autocomplete Feature Of The Textbox Based On Ajax And Web Service*. Journal of Computers. (8) 9 : 2199-2201.

UNNES
UNIVERSITAS NEGERI SEMARANG