Jurnal Teknik Elektro Vol. 7 No. 1
Januari - Juni 2015

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

Khairun Nisa Meiah Ngafidin dan Hari Wibawanto

*Jurusan Teknik Elektro, Fakultas Teknik, Universitas Negeri Semarang, Indonesia*
*nisameiah@gmail.com*

**Abstrak— Penelitian ini dilakukan untuk mengimplementasikan fitur autocomplete dan algoritma levenshtein distance pada apllikasi KBBI dan untuk mengetahui efektivitas penggunaannya dalam fitur pencarian aran kata. Metode pengembangan software yang digunakan adalah dengan menggunakan metode waterfall, yang terdiri dari lima bagian yaitu requirement definitions, system and software design, implementation and unit testing, integration and system testing, dan operation and maintenance.Hasil penelitian yang didapat dari pengujian black box terhadap kemunculan autocomplete adalah muncul untuk setiap kata yang diinputkan. Lalu untuk pengujian dengan algoritma levenshtein distance, saran sudah bisa muncul meskipun tidak semua saran sesuai dengan yang diharapkan dan untuk pengujian terhadap keseluruhan sistem aplikasi dihasilkan keluaran yang valid untuk setiap menu yang diuji. Pengujian keefektifan terhadap efektifitas implementasi autocomplete pada aplikasi adalah sebesar 84.615 % yang berarti fitur ini sangat efektif. Dan untuk levenshtein distance adalah sebesar 76.04 % yang berarti efektif untuk digunakan di aplikasi KBBI. Saran yang dapat diberikan dalam penelitian ini adalah sebaiknya dilakukan penambahan menu pencarian kata dan ungkapan daerah, kata dan ungkapan asing, dan sinonim dan akronim agar kamus digital ini lebih lengkap seperti versi cetaknya.**

**Kata kunci— Letakkan kata kunci Anda di sini, kata kunci dipisahkan dengan koma.**

## I. PENDAHULUAN

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 baik itu menggunakan *wifi* maupun sambungan data dari *provider*.Kamus daring di *android* berarti bahwa basis data atau penyimpanan data kamus disimpan di *serveronline* 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 *serveronline*. 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. *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). 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).

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).

Selain itu 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. Perhitungan *levenshtein 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*(Dwitiyastuti, 2010).

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

a. Operasi penambahan 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 pengubahan 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".

Berikut ini merupakan rumus untuk menghitung kemiripan kata menurut Riya Mary Abraham (2014 : 227) :

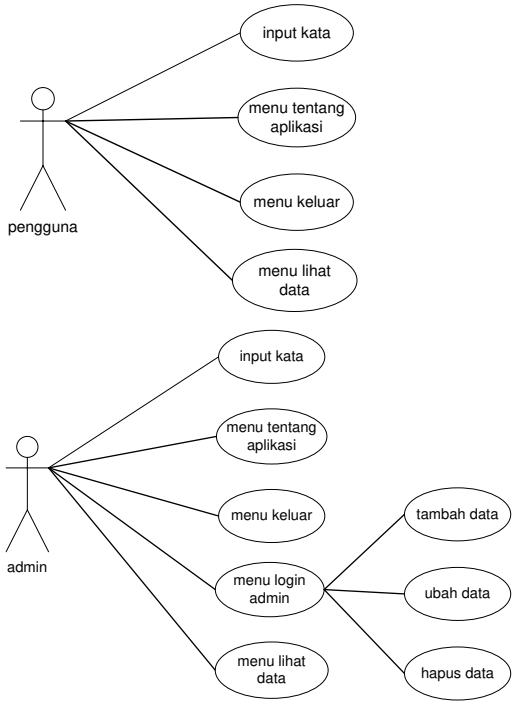
$$\text{Sim} = 1 - \frac{d}{\max of\ 2\ strings}$$

Keterangan :
Sim = similarity / kemiripan
d = nilai edit distance
max of 2 *string* = jumlah maksimal karakter dari kedua kata

## II. METODE

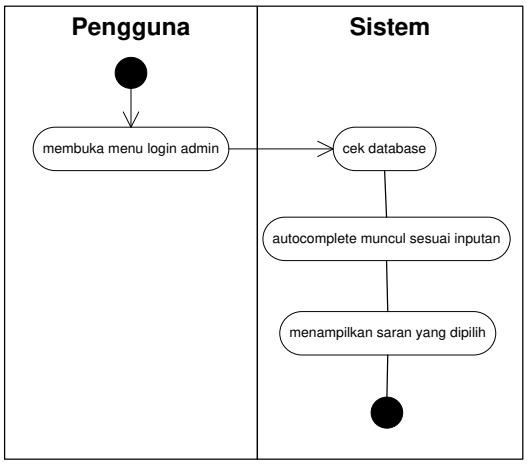
Dalam proses pembuatan aplikasi KBBI ini digunakan suatu model pengembangan *software* yaitu *waterfall* atau disebut juga dengan sekuensial linier. Menurut (Roger S. Pressman, 1997:37) *waterfall* atau sekuensial linier merupakan suatu pendekatan pada perkembangan perangkat lunak yang sistemik dan sekuensial. Dimana model ini dikatakan model konvensional atau model lama, yang prosesnya hanya bisa dilakukan jika proses sebelumnya telah selesai, dan jika belum selesai maka belum bisa dilakukan proses yang selanjutnya, sehingga dikatakan juga dengan metode yang sistematis dan berurutan. Proses pengembangan *software* ini dimulai dari mendefinisikan kebutuhan, perancangan perangkat lunak dan sistem, implementasi, pengujian, serta operasi dan pemeliharaan.

1. Definisi Kebutuhan
   Dalam proses pertama metode *waterfall* yang harus dilakukan adalah menganalisis dan melihat definisi persyaratan yang dibutuhkan dalam pembuatannya. Hal ini bisa dilakukan dengan mengumpulkan semua bahan-bahan yang dibutuhkan seperti data penunjang *software*, dan lain sebagainya yang kemudian dianalisis dan didefinisikan kebutuhan *software* tersebut.
2. Perancangan perangkat lunak dan sistem
   Pada proses ini dilakukan desain untuk tampilan secara keseluruhan. Proses ini hanya dapat dilakukan ketika definisi kebutuhan selesai dikerjakan dan diketahui hasilnya, lalu dibuatlah desain sistem dan tampilannya. Berikut ini merupakan perancangan sistem dari aplikasi KBBI dengan menggunakan use case diagram dan activity diagram :
   1) *Use Case Diagram*
      Proses perancangan *software* untuk KBBI ini dimulai dengan merancang bagaimana sistem ini akan berjalan. Dan agar lebih mudah, dibawah ini akan digambarkan dalam bentuk UML, yaitu *use case* maupun *activity diagram.*

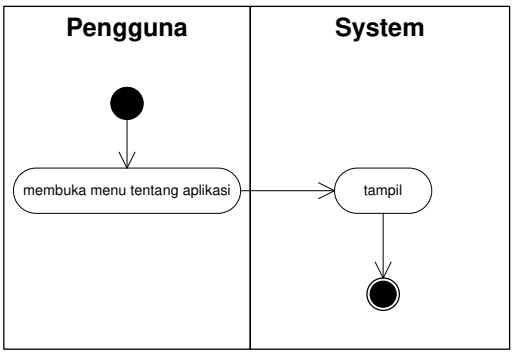



Gambar 1. *Use Case Diagram*

   2) *Activity Diagram*
      Activity diagram merupakan suatu urutan aktivitas yang menggambarkan suatu proses. Activity diagram yang dibuat ini berdasarkan atas use case yang telah dibuat. Activity diagram yang ada pada aplikasi KBBI ditunjukkan pada Gambar 2 dan Gambar 3.

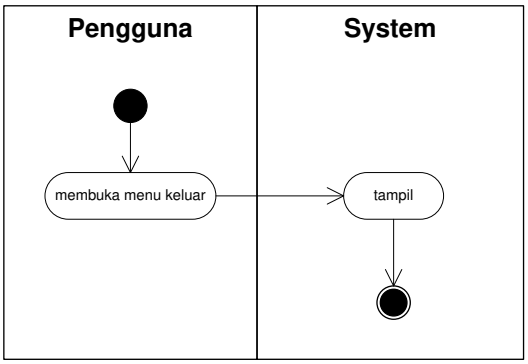



Gambar 2. *Activity Diagram* Proses *Autocomplete*

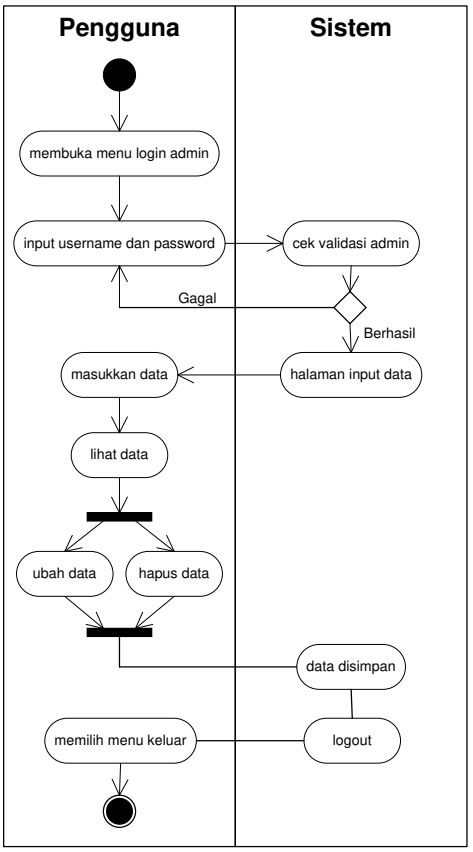



Gambar 3. *Activity Diagram* Membuka Menu Tentang Aplikasi

Pada diagram aktivitas diatas, pengguna hanya melakukan satu aktivitas saja yaitu membuka menu tentang aplikasi. Setelah menu tentang aplikasi diklik, maka akan muncul tampilan mengenai deskripsi singkat mengenai aplikasi KBBI, dan setelah selesai membaca maka dapat kembali ke menu pencarian dengan menekan button icon aplikasi KBBI.

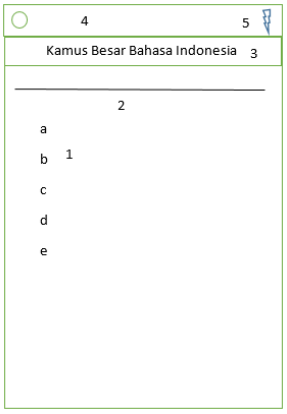



Gambar 4. *Diagram* Menutup Aplikasi




Gambar 5. *Use Case Diagram* dari Menu *Login Admin*

Setelah perancangan sistem dibuat, langkah selanjutnya adalah membuat rancangan tampilan. Tampilan dari perancangan perangkat lunak aplikasi KBBI ditunjukkan pada Gambar 6.




Gambar 6. Tampilan Halaman Utama

3. Implementasi
   Dalam proses ini *software* yang telah dirancang dan telah ditetapkan persyaratan dan analisa kebutuhannya, lalu desain telah selesai dibuat, maka langkah selanjutnya adalah pengimplementasian dari hal-hal diatas. Untuk itu berikut ini merupakan implementasi *software* yang dilakukan :
   a) Analisis perangkat lunak
      Perangkat lunak yang digunakan dalam pengimplementasian aplikasi KBBI adalah sistem operasi android dengan versi 4.0 keatas, atau Ice Cream Sandwich, JellyBean, Kitkat, maupun Lollipop.
   b) Analisis perangkat keras
      Perangkat keras yang digunakan dalam pengimplementasian aplikasi KBBI ini diantaranya yaitu :
      - Processor : CPU 800 MHz Cortex-A5
      - Memori : Internal 2 GB
      - RAM : 512 MB
      - Layar : TFT *Capacitive Touchscreen*
4. Pengujian
   Proses ini merupakan proses dimana *software* yang telah dibuat kemudian diuji apakah sudah sesuai dengan yang diharapkan atau belum. Pengujian dilakukan dengan menggunakan metode *black box.*
5. Pengoperasian dan pemeliharaan
   Tahap terakhir dari urutan proses dengan menggunakan metode *waterfall* adalah dengan menjalankan dan kemudian dilakukan pemeliharaan. Aplikasi dijalankan berarti aplikasi tetap digunakan dan disebarluaskan untuk pengguna yang membutuhkan aplikasi KBBI ini. Dan untuk pemeliharaan bisa dilakukan untuk menjaga aplikasi tetap *up to date*, dengan menambahkan kata baru pada aplikasi jika ditemukan.

## III. PENGUJIAN

Setelah dilakukan pengujian terhadap aplikasi KBBI, maka dibawah ini merupakan hasil yang didapat dari proses pengujian dengan metode *black box*.

1. Pengujian Aplikasi KBBI
   Hasil pengujian yang dilakukan dengan metode *black box* pada sistem aplikasi KBBI Ditunjukkan pada Tabel 1.
2. Pengujian Autocomplete
   Pengujian *autocomplete* merupakan pengujian untuk mengetahui apakah *autocomplete* dapat memunculkan pilihan kata atau tidak. Hasil pengujian dari *autocomplete* di aplikasi KBBI ditunjukkan pada Tabel 2.
3. Pengujian Kinerja
   Pengujian kinerja merupakan pengujian pada sistem yang diujikan untuk dapat memberikan saran kata yang dicari berdasarkan kata yang mirip dalam basis data. Hasil pengujian pemberian saran dengan algoritma *levenshtein distance* ditunjukkan pada Tabel 3.

TABEL I. HASIL PENGUJIAN SISTEM APLIKASI KBBI

| **No** | **Hal yang diuji** | **Butir uji** | **Hasil pengujian** |
|---|---|---|---|
| 1 | Menu tentang aplikasi | Menampilkan deskripsi singkat aplikasi | Valid |
| 2 | Menu keluar | Menampilkan kotak dialog dengan pernyataan memilih iya atau tidak | Valid |
| 3 | Menu login admin | Dapat login dengan username dan password yang disediakan | Valid |
| | | Dapat menambahkan data dan menyimpannya | Valid |
| | | Dapat mengubah data | Valid |
| | | Dapat menghapus data | Valid |
| | | Dapat melihat data secara keseluruhan | Valid |
| | | Dapat logout dari menu admin | Valid |
| 4 | Menu lihat data | Dapat menampilkan daftar kata yang ada | Valid |
| 5 | *Autocomplete* | Dapat menampilkan autocomplete | Valid |
| 6 | Pemberian saran | Dapat menampilkan saran | Valid |

TABEL II. HASIL PENGUJIAN *AUTOCOMPLETE*

| No | Kata inputan | Muncul/ Tidak | Saran yang muncul |
|---|---|---|---|
| 1 | A | Muncul | a, a-, aa, ab |
| 2 | Ba | Muncul | Bakda pisang, bakda seluang, bakdahu |
| 3 | Ca | Muncul | Ca, caba, cabai, cabak |
| 4 | Da | Muncul | Dahak, daham, dahan, dahar |
| 5 | Ef | Muncul | Efek, efektif, efektivitas, efelis |
| 6 | Fa | Muncul | Fase, faset, fasia, fasid |
| 7 | Ge | Muncul | Gen, gena, genah, genahar |
| 8 | Ha | Muncul | Habenula, habib, habibullah, habis |
| 9 | In | Muncul | Ina, inadaptabilitas, inai, inaktivator |
| 10 | Je | Muncul | Je, jebab, jebah, jebai |
| 11 | Ka | Muncul | Kab, kaba, kabab, kabaea |
| 12 | Le | Muncul | Leak, lebah, lebai, lebak |
| 13 | Ma | Muncul | Mah, maha-, mahabah, mahabharata |
| 14 | Na | Muncul | Nara, narapati, narapidana, narapraja |
| 15 | Ok | Muncul | Oke, oker, oklokrasi, oklusi |
| 16 | Pasca | Muncul | Pasca-, pascabedah, pascacedera, pascadoktoral |
| 16 | Pasca | Muncul | Pasca-, pascabedah, pascacedera, pascadoktoral |
| 17 | Ra | Muncul | Randa, randa-rondo, randah, randai |
| 18 | Semi | Muncul | Semi-, semiang, semidiurnal, semifinal |
| 19 | Ta | Muncul | Tasak, tasalsul, tasamuh, tasaruf |
| 20 | Uri | Muncul | Urik, urinalisis, urine, uring |
| 21 | Va | Muncul | Vagina, vaginal, vaginektomi |
| 22 | Wa | Muncul | Waadat, wabah, wabakdahu, wabarakatuh |
| 23 | Xe | Muncul | Xenofobia, xenoglosia, xenoglosofobia |
| 24 | Ya | Muncul | Yalusi, yatuan, yang, yang-yang |
| 25 | Ze | Muncul | Zelofobia, zelot, zen, zendavista |

TABEL III. HASIL PENGUJIAN PEMBERIAN SARAN

| Kategori | Kata | Kata yang diinginkan | Saran yang muncul |
|---|---|---|---|
| Penggantian satu huruf | Faseh | Fasih | Fasih |
| Penyisipan satu huruf | Tasyaruf | Tasaruf | Tasaruf |
| Penghapusan satu huruf | Jebt | Jebat | Jebat |
| Penukaran dua huruf | Genecl | Gencel | Gencel |
| Penggantian dua huruf | Zibro | Zebra | Zebra |
| Kesalahan lebih dari 1 huruf | Habtasi | Habituasi | Habituasi |

## IV. PEMBAHASAN

Penerapan fitur *autocomplete* pada aplikasi KBBI ini memberikan kemudahan pada pengguna dalam proses pencarian kata Bahasa Indonesia. Dengan hasil uji efektivitas yang berjumlah 84,615 %, yang berarti bahwa fitur *autocomplete* ini sangat efektif untuk digunakan dalam implementasi kamus. Namun selain kamus, fitur *autocomplete* ini juga sudah banyak digunakan dalam aplikasi-aplikasi serupa, baik itu berbasis web maupun android. Seperti pada penelitian yang dilakukan oleh Zhiqiang Yao (2013) yang mengimplementasikan fitur *autocomplete* pada *textbox* yang berdasar *Ajax* dan *Web Service*. Hasil pengujian yang telah dilakukan oleh Zhiqiang Yao didapat kesimpulan bahwa efektivitas penginputan meningkat, kesalahan pengejaan bisa dihindari, dan kemampuan interaksi manusia pada komputer menguat.

Begitu juga dengan algoritma *levenshtein distance*, sudah mulai banyak peneliti yang mengembangkan aplikasi menggunakan algoritma tersebut untuk berbagai macam keperluan. Pada penelitian yang dilakukan oleh (Chen, 2014) yang mengimplementasikan algoritma *levenshtein distance* untuk pendeteksian *e-mail hoax*. Dengan algoritma ini sistem dapat memprediksi dengan baik dengan nilai 0,96, namun memiliki kekurangan belum bisa mengidentifikasi e-mail yang asli.Pada implementasi *levenshtein distance* di aplikasi KBBI juga terdapat beberapa kasus dimana saran yang muncul tidak sesuai yang diharapkan. Hal ini bisa karena jumlah basis data yang tidak banyak sehingga penghitungan jarak antar kata hanya sedikit. Namun tetap dalam pengujian efektivitas algoritma ini pada aplikasi KBBI bernilai 76,04 % yang berarti efektif untuk digunakan.

Pada penelitian yang dilakukan oleh Riya Mary Abraham (2014) algoritma *levenshtein distance* diimplementasikan dalam pencarian di *cloud environment*. Hasil yang didapat dibandingkan dengan pencarian tradisional, algoritma ini sedikit lebih unggul. Sehingga tidak salah jika algoritma ini efektif untuk digunakan dalam berbagai aplikasi atau kegunaan lain.Dari penelitian oleh Riya Mari Abraham tersebut algoritma *levenshtein distance* memiliki nilai yang tinggi dalam memprediksi kemiripan kata yang dimasukkan pengguna. Jika dalam aplikasi KBBI tidak terdapat perbandingan penggunaan *autocomplete* dengan algoritma *levenshtein distance*, karena keduanya saling bekerja sama dalam proses pencarian kata. Namun jika dilihat dari hasil penilain responden mengenai efektivitas keduanya, nilai tertinggi terdapat pada *autocomplete* yang berarti sangat efektif, dan algoritma *levenshtein distance* yang mempunyai hasil efektif untuk digunakan dalam proses pencarian di aplikasi KBBI.

Penelitian yang dilakukan oleh Aouragh Si Lhoussain (2015) tentang implementasi *levenshtein distance* pada pengoreksian ejaan kontekstual. Aouragh melakukan pengenalan metode *bi-gram* pada algoritma *levenshtein distance*. Hasilnya didapatkan bahwa kombinasi tersebut dapat meningkatkan kepuasan dalam solusi metode tersebut. Hal ini pun sama dengan hasil pengujian pada *autocomplete* dan algoritma *levenshtein distance*, dimana responden setuju jika fitur dan algoritma tersebut efektif untuk diimplementasikan pada aplikasi KBBI.

## V. SIMPULAN

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 %.

## REFERENSI

[1] Abraham, Riya Mary. 2014. Use Of Edit Distance Algorithm To Search A Keyword In CloudEnvironment. International Journal Of Database Theory And Application. (7) 6 : 223-232.

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

[3] Anonim. Tanpa Tahun. AutocompleteTextView. http://developer.android.com/. Diakses 19 Oktober 2015.

[4] 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.

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

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

[7] 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.

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

[9] Dixon, Helen. 2007. Excel 2007 : Beyond the Manual. Springer-Verlag : New York.

[10] Dwitiyastuti, Rachmania Nur dkk. Tanpa Tahun. Pengoreksi Kesalahan Ejaan Bahasa Indonesia Menggunakan Metode Levenshtein distance. http://download.portalgaruda.org/. Diakses 15 Februari 2015.
[11] Fatta, Hanif Al. 2007. Analisis dan Perancangan Sistem Informasi untuk Keunggulan Bersaing Perusahaan dan Organisasi Modern. ANDI : Yogyakarta.
[12] Geisler, Reiner. 2015. Industrial Software Applications. http://dnb.dnb.de. Diakses 23 Oktober 2015.
[13] 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.
[14] Haryanto, Edy Victor. 2011. Rancang Bangun Prototype Mesin Pencari String Menggunakan Metode Fuzzy String Matching. Konferensi Nasional Sistem dan Informatika, Bali.
[15] Hyvonen, Eero., Eetu Makela. 2006. Semantic Autocompletion. Semantic Computing Research Group (SeCo). http://www.seco.tkk.fi/. Diakses 10 Februari 2015.
[16] Ida, Winarti. 2010. Pengaruh Area Hotspot (Wi-Fi) Bagi Pemenuhan Kebutuhan Informasi Pemustaka di Kantor Perpustakaan Daerah Kabupaten Jepara. Undergraduate thesis, Faculty of Humanities.
[17] 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.
[18] 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.
[19] Lhoussain, Aouragh Si. 2015. Adaptating The Levenshtein Distance To Contextual Spelling Correction. International Journal Of Computer Science And Application. (12) 1 : 127-133.
[20] 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.
[21] Oei, Istijanto. 2005. Riset : Sumber Daya Manusia. Gramedia : Jakarta. Pressman, Roger S, Ph.D. 1997. Rekayasa Perangkat Lunak. ANDI and McGraw-Hil Book : Yogyakarta.
[22] Pusat Bahasa. 2008. Kamus Besar Bahasa Indonesia Pusat Bahasa. Gramedia Pustaka Utama : Jakarta.
[23] Rahardja, Yani,.dkk. 2008. Analisis dan Perancangan Mobile-Banking dengan Menggunakan UML. Jurnal Teknologi Informasi-Aiti. (5) 2 : 101-200.
[24] 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.
[25] Supriyanto, Dodit., Rini Agustina. 2012. Pemrograman Aplikasi Android : Step by Step Membuat Aplikasi Android untuk Smartphone dan Tablet. Mediakom: Yogyakarta.
[26] Suryawinata, Z & S.Hariyanto. 2003. Translation: Bahasa dan Penuntun Praktis Menerjemahkan. Yogyakarta: Kanisius.
[27] Wicaksono, Yogi. 2008. Membangun Bisnis Online Dengan Mambo. Elex Media Komputindo: Jakarta.
[28] Wiyanto, Asul., dkk. 2005. Mampu Berbahasa Indonesia. Grasindo: Jakarta.
[29] Yao, Zhiqiang. 2013. Implementation Of The Autocomplete Feature Of The Textbox Based On Ajax And Web Service. Journal of Computers. (8) 9 : 2199-2201.

Jurnal Teknik Elektro Vol. 7 No. 1
Januari - Juni 2015

# Analisis Aliran Daya Sistem Tenaga Listrik pada Bagian *Texturizing* di PT Asia Pasific Fibers Tbk Kendal menggunakan *Software* ETAP Power Station 4.0

Adib Gustian Nigara dan Yohanes Primadiyono

*Jurusan Teknik Elektro, Fakultas Teknik, Universitas Negeri Semarang, Indonesia*

*adiebgustian@gmail.com*

**Abstrak— Analisis Aliran Daya Listrik (Load Flow) adalah suatu studi untuk merencanakan dan mengetahui besarnya daya dalam suatu sistem tenaga listrik. Dalam perkembangannya, industri membutuhkan tenaga listrik yang besar dan menggunakan peralatan listrik sebagai alat produksi. Manfaat dari adanya analisis aliran daya listrik adalah untuk mengetahui besarnya daya dalam sistem tenaga listrik apakah masih memenuhi batas-batas yang telah ditentukan, serta untuk mengetahui besar *Losses* yang ada, dan untuk memperoleh kondisi mula pada perencanaan sistem yang baru. Studi analisis aliran beban ini mengambil contoh pada implementasi sistem tenaga listrik di PT. Asia Pasific Fibers Tbk Kendal, dengan karakteristik beban terpusat (*lumped load*). Analisis aliran daya diawali menghitung daya aktif dan daya reaktif pada setiap simpul (*bus*) terpasang, pembebanan pada transformator, pembebanan pada saluran atau penghantar, nilai rugi daya (Losses), jatuh tegangan sistem, dan aliran daya pada jaringan sistem tenaga listrik terpasang. Dari hasil perhitungan aliran daya berbantuan program ETAP (*Electrical Transient Analyzer Program*) dapat disimpulkan bahwa sistem jaringan listrik sudah baik. Hasil yang diperoleh adalah selisih rugi daya aktif dan rugi daya reaktif pada Bus Beban 4 terlalu besar. Sedangkan jatuh tegangan masih memenuhi standar menurut hasil *Text Report* pada ETAP.**

**Kata kunci—** ***Load Flow, Losses, ETAP***

## I. PENDAHULUAN

Dalam dunia industri, listrik sudah menjadi kebutuhan primer sebagai sumber energi utama untuk menjalankan semua alat dan mesin yang ada di industri. Di PT Asia Pasific Fibers, khususnya pada bagian Texturizing, untuk memenuhi kebutuhan listrik, energi listrik diambil dari pasokan listrik Perusahaan Listrik Negara (PLN). Seiring berjalannya waktu, bagian *Texturizing* pun terus melakukan pengembangan khususnya di sektor pembangunan, yang berarti bertambah pula jumlah beban yang harus ditanggung. Akibatnya, desain konfigurasi awal dari sistem jaringan kelistrikan yang awalnya baik dan mampu melayani beban dengan baik, bisa jadi menjadi tidak sesuai lagi dengan keadaan pembebanan saat ini. Untuk itu, perlu dilakukan analisis aliran daya untuk mengetahui kondisi secara keseluruhan dari sistem tenaga listrik pada bagian *Texturizing* di PT Asia Pasific Fibers saat ini.

Analisis aliran daya dalam sistem tenaga listrik merupakan analisis yang mengungkapkan kinerja suatu sistem tenaga listrik dan aliran daya (nyata dan reaktif) untuk keadaan tertentu ketika sistem bekerja. Hasil utama dari aliran daya adalah besar dan sudut fasa tegangan pada setiap saluran (bus), daya nyata dan daya reaktif yang ada pada setiap saluran. Hasil analisis aliran daya dapat digunakan untuk mengetahui besarnya *losses* (rugi daya dan tegangan), alokasi daya reaktif dan kemampuan sistem untuk memenuhi pertumbuhan beban.

Perhitungan aliran daya untuk sistem tenaga listrik pada bagian *Texturizing* di PT Asia Pasific Fibers secara manual akan sangat rumit, oleh sebab itu dalam penelitian ini digunakan *software* komputer untuk mempermudah dan mempercepat dalam proses perhitungan aliran daya. ETAP (*Electrical Transient Analisis Program*) Power Station merupakan salah satu *software* yang dapat digunakan untuk perhitungan aliran daya pada sistem tenaga listrik. Dengan menggunakan *software* ETAP Power Station 4.0 akan dapat menganalisis sistem tenaga listrik yang sangat luas (Agung, 2009 dalam Dhimas, 2014 : 2).

Penelitian ini bertujuan untuk mengetahui: 1) besar nilai aliran daya yang meliputi daya aktif, daya reaktif dan daya semu yang ada pada bagian *Texturizing* di PT Asia Pasific Fibers, 2) besar nilai rugi daya (*losses*) yang terdapat dalam sistem tenaga listrik pada bagian Texturizing di PT Asia Pasific Fibers, 3) besar nilai tegangan dan jatuh tegangan pada setiap bus yang ada pada bagian Texturizing di PT Asia Pasific Fibers.