Software Aplikasi Pengolah Kata (Word Processor) Dengan Fasilitas Pemeriksa Ejaan Dan Thesaurus Berbahasa Indonesia

Agus Zainal Arifin, Arif Bramantoro
Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember (ITS) – Surabaya
Kampus ITS, Jl. Raya ITS, Sukolilo-Surabaya 60111
Tel. +62 31 5939214, Fax + 62 31 5939363
Email: agusza@its-sby.edu ; bram@its-sby.edu

ABSTRAK

Kebutuhan akan adanya sistem pengolahan kata elektronis, dalam fungsinya sebagai penyunting kata/dokumen, dirasakan sangat penting. Perangkat lunak pengolah kata seperti Microsoft Word dan Amipro telah memberikan fasilitas pendukung untuk pemeriksaan ejaan dan pemilihan kata (dengan thesaurus). Akan tetapi, fasilitas tersebut hanya diaplikasikan untuk Bahasa Inggris. Sehingga perlu adanya usaha-usaha untuk mengaplikasikan fasilitas tersebut dalam Bahasa Indonesia.

Diantara usaha tersebut adalah pembuatan perangkat lunak untuk memeriksa validitas suatu kata, yang perlu dilakukan pengecekan kata dalam kamus untuk menentukan apakah kata tersebut terdapat dalam kamus atau tidak. Dalam aplikasi pengecekan kata itu diperlukan suatu kamus elektronis. Struktur kamus elektronis sangat menentukan waktu pencarian kata dan memori yang dibutuhkan untuk menyimpan kata-kata dasar pada saat runtime. Struktur digital Tree memberikan keduanya. Selain itu perlu juga dipertimbangkan informasi frekuensi kemunculan suatu kata dalam Bahasa Indonesia.

Dengan informasi ini diharapkan semakin sering suatu kata digunakan dalam Bahasa Indonesia, maka makin cepatlah kata itu ditemukan dalam kamus. Sehingga secara keseluruhan akan menghemat waktu pencarian semua kata dalam file sampel.

Kata Kunci : *Pengolah Kata, Thesaurus, Ejaan, Digital Tree, Kamus Bahasa*

1. PENDAHULUAN

Aplikasi Pengolahan Kata adalah sebuah window tempat pengguna dapat memasukkan dan menyunting teks. Teks dapat diisi dengan karakter dan pemformatan paragraf dan dapat dimasukkan objek-objek OLE. Kontrol-kontrol yang ada di dalam aplikasi ini menyediakan programming interface ditambah dengan implementasi komponen-komponen user interface untuk melakukan operasi-operasi pemformatan yang disediakan untuk pengguna.

Kontrol-kontrol aplikasi ini mendukung hampir seluruh pesan-pesan dan notifikasi yang digunakan dengan kontrol multiline edit. Pesan-pesan yang dikirim antar objek didalam aplikasi ini digunakan untuk melakukan operasi-operasi, seperti pemformatan teks, pencetakan, dan penyimpanan. Aplikasi ini dapat memproses pesan-pesan notifikasi untuk memonitor kejadian-kejadian (*event*) didalam tiap objeknya. Contohnya, aplikasi dapat memproses notifikasi untuk memfilter masukan dari keyboard dan mouse, untuk mengijinkan atau melarang perubahan untuk memproteksi teks atau mengubah kontrol yang dibutuhkan untuk memenuhi isinya.

Kebutuhan akan adanya sistem pengolahan kata elektronis, dalam fungsinya sebagai penyunting kata/dokumen, dirasakan sangat penting. Perangkat lunak pengolah kata seperti *Microsoft Word* dan *Amipro* telah memberikan fasilitas pendukung untuk pemeriksaan ejaan dan pemilihan kata (dengan thesaurus). Dalam aplikasi pengecekan kata itu diperlukan suatu kamus elektronis. Struktur kamus elektronis sangat menentukan waktu pencarian kata dan memori yang dibutuhkan untuk menyimpan kata-kata dasar pada saat runtime. Struktur digital Tree memberikan keduanya. Selain itu perlu juga dipertimbangkan informasi frekuensi kemunculan suatu kata dalam Bahasa Indonesia. Dengan informasi ini diharapkan semakin sering suatu kata digunakan dalam Bahasa Indonesia, maka makin cepatlah kata itu ditemukan dalam kamus. Sehingga secara keseluruhan akan menghemat waktu pencarian semua kata dalam file sampel.

Thesaurus adalah bentuk yang berharga dalam sistem pencarian informasi. Sebuah thesaurus akan menyediakan daftar kata yang tepat dan terkontrol yang berguna dalam mengkoordinasikan pengindeksan maupun pencarian dokumen. Thesaurus telah digunakan dalam dunia informasi untuk memecahkan masalah ketidakkonsistenan pada pengindeksan dokumen, dan juga dapat digunakan oleh pencari dalam memformulasi ulang strategi pencarian yang tepat jika diperlukan.

Pembangunan thesaurus berbahasa Indonesia disini, didasari oleh kebutuhan akan pencarian informasi berbahasa Indonesia. Pembangunan thesaurus dapat dilakukan dengan dua cara yaitu dengan cara manual atau dengan cara otomatis.

Pembangunan Thesaurus secara otomatis dilakukan dengan melakukan analisa terhadap kemunculan pasangan kata(co-word) dalam kumpulan dokumen. Oleh karena itu diperlukan pembelajaran(learning) untuk mengelompokkan kata berbahasa Indonesia dengan tingkat kemiripan tertentu berdasar frekuensi kemunculan pasangan kata dalam koleksi dokumen.

Kemampuan thesaurus dalam menemukan kata akan sangat tergantung dengan ruang lingkup pembahasan, dimana eksekusi pada ruang lingkup hukum akan sangat berbeda maknanya dibanding dengan eksekusi pada ruang lingkup teknologi komputer. Kumpulan dokumen yang digunakan dalam pembangunan thesaurus diambil dari berbagai sumber seperti berita, artikel, dan sebagainya.

Tujuan dari penelitian ini adalah membangun perangkat lunak pengolah kata yang mampu melakukan pengecekan terhadap ejaan dalam Bahasa Indonesia, serta memiliki fasilitas untuk menentukan thesaurus dari sebuah kata yang tertulis dalam text processor.

Permasalahan yang muncul dari pembuatan perangkat lunak ini antara lain :

- Bagaimana membangun daftar kata baru yang ditemukan pada sebuah dokumen koleksi menjadi sebuah kamus kata.
- Bagaimana menghitung tingkat kemiripan antar kata yang terdapat dalam kamus.
- Bagaimana mengorganisasikan kata yang ada menjadi sebuah thesaurus.

2. RINGKASAN TEORI

Produk yang dihasilkan pada riset ini adalah sebuah aplikasi pengolah kata. Berikut ini disajikan beberapa konsep dasar yang digunakan pada pembangunan perangkat lunak ini [4][6].

2.1. Algoritma Stemming

Algoritma ini didahului dengan pembacaan tiap kata dari file sampel. Sehingga input dari algoritma ini adalah sebuah kata yang kemudian dilakukan:

1. Pemeriksaan semua kemungkinan bentuk kata. Setiap kata diasumsikan memiliki 2 awalan / prefiks dan 3 akhiran / sufiks. Sehingga bentuknya menjadi :
   Prefiks 1 + Prefiks 2 + Kata Dasar + Sufiks 3 + Sufiks 2+ Sufiks 1
   Seandainya kata tersebut tidak memiliki imbuhan sebanyak imbuhan di atas, maka imbuhan yang kosong diberi tanda x untuk prefiks dan diberi tanda xx untuk sufiks. Untuk mewujudkannya maka dibuatlah struktur data untuk menampung setiap kata yang bentuknya sebagai berikut :

```
enum awalan_t {AwalanError=0,x,
     me, pe, be, di, se, ke, te,
     mem=100, men, per, pem, ber, ter,
pen,
     ber_luluh, ter_luluh, per_luluh,
     mem_luluh, pem_luluh, men_luluh,
pen_luluh, meny=200, peny, meng,
     meng_luluh, peng_luluh, peng
    };

enum akhiran_t {AkhiranError=0, i, kan,
an, ku, mu, lah, pun,  nya, kah, xx};

struct arrkata_t {
 enum awalan_t p1,p2;
 char kd[30];
 enum akhiran_t s3,s2,s1;
 };
```

2. Dengan struktur data di atas, maka langkah awal pemotongan bisa dari mana saja. Dalam hal ini pemotongan dilakukan secara berurutan sebagai berikut :

   a. Awalan I, hasilnya disimpan pada p1
   b. Awalan II, hasilnya disimpan pada p2
   c. Akhiran I, hasilnya disimpan pada s1
   d. Akhiran II, hasilnya disimpan pada s2
   e. Akhiran III, hasilnya disimpan pada s3

   Pada setiap tahap pemotongan di atas diikuti dengan pemeriksaan di kamus apakah hasil pemotongan itu sudah berada dalam bentuk dasar. Kalau pemeriksaan ini berhasil maka proses dinyatakan selesai dan tidak perlu melanjutkan proses pemotongan imbuhan lainnya.

3. Namun jika sampai pada pemotongan akhiran III, belum juga ditemukan di kamus, maka dilakukan proses kombinasi. Kata dasar yang dihasilkan dikombinasikan dengan imbuhan-imbuhannya dalam 12 konfigurasi berikut :

   a. Kata Dasar
   b. Kata Dasar + Akhiran III
   c. Kata Dasar + Akhiran III + Akhiran II
   d. Kata Dasar + Akhiran III + Akhiran II + Akhiran I
   e. Awalan I + Awalan II + Kata Dasar
   f. Awalan I + Awalan II + Kata Dasar + Akhiran III
   g. Awalan I + Awalan II + Kata Dasar + Akhiran III + Akhiran II
   h. Awalan I + Awalan II + Kata Dasar + Akhiran III + AkhiranII + AkhiranI
   i. Awalan II + Kata Dasar
   j. Awalan II + Kata Dasar + Akhiran III
   k. Awalan II + Kata Dasar + Akhiran III + Akhiran II

l. Awalan II + Kata Dasar + Akhiran III + Akhiran II + Akhiran I

2.2 Vector Model

Dokumen dan term haruslah direpresentasikan oleh vector space. Misalkan t1, t2, …, tn menyatakan term yang digunakan untuk mengindex database yang terdiri dari dokumen D1, D2, … Dn, maka dokumen Di dinyatakan dengan Di = (ai1, ai2, … ain), dimana aij = bobot term tj dalam dokumen Di.
Jadi nilai tiap elemen dalam vector tersebut diwakili oleh bobot termnya. Mei Kobayashi [8] memberikan alternatif pembobotan term, antara lain :

- Binary weighting, dimana w(t,d) = 0, jika tf(t,d) = 0, dan w(t,d) = 1, jika tf(t,d) > 0.
- Term frequency weighting, dimana w(t,d) = tf(t,d).
- Log Entropy weighting, dimana w(t,d) = log2(tf(t,d) + 1)

Ada juga fungsi pembobotan global yang cukup mudah dan sederhana, misalnya :

- Normal : $G(t) = (1/\sum_j tf(t,d))^{1/2}$
- GdIdf, dimana : $G(t) = gft / dft$ (gft adalah
- frekuensi global term t, dft = nt)
- Idf, dimana : $G(t) = log2$ (Jumlah Dokumen / dft) + 1
- Entropi, dimana : $G(t) = 1 - \sum_j [(ptj\ log2(ptj)) / log2(Jumlah\ Dokumen)]$
  $ptj = tf(t,d) / gft$

Sedangkan dalam penelitian ini, untuk pembobotan term (term weighting), kami menggunakan TF-IDF. Metode ini dibahas penggunaannya oleh Alberto [5] dan Yiming Yang [9]. Rumusnya adalah sebagai berikut :

$$w(t,d) = tf(t,d)\ x\ log(N / nt)$$

dimana tf(t,d) merupakan term frequency yakni jumlah term t dalam dokumen d. sedangkan N adalah jumlah seluruh dokumen dan nt adalah jumlah dokumen yang memiliki term t.

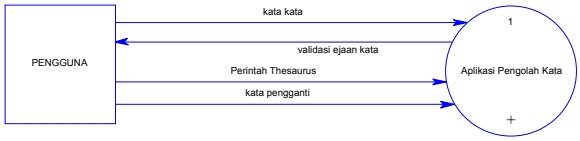
Lebih lanjut Salton [2] menjelaskan alasan penggunaan metode ini. Ternyata sekalipun term frequency banyak digunakan, namun ia hanya mendukung Recall. Sedangkan Precision akan lebih meningkat bila vector bobot tersebut menggunakan term yang jarang muncul pada koleksi dokumen. Tentunya term demikian akan diharapkan mampu mengelompokkan sejumlah dokumen yang memuatnya, sehingga berbeda dengan seluruh anggota koleksi dokumen lain yang tidak memilikinya. Kriteria ini dapat diakomodasi dengan menghitung log (N / nt) seperti di atas. Sedangkan alasan digabungnya kedua metode tersebut adalah dimaksudkan agar mampu meningkatkan baik Recall maupun Precision sekaligus. Sehingga kriteria term yang paling tepat adalah term yang sering muncul dalam dokumen secara individu, namun jarang dijumpai pada dokumen lainnya.

3. PERANCANGAN PERANGKAT LUNAK

Dalam mengerjakan perangkat lunak ini dilakukan beberapa proses, yaitu pengolahan data, proses koneksi web server dengan map server, pemecahan parameter URL, menentukan batasan, penentuan lokasi, dan pembuatan halaman HTML.

3.1 Perancangan Proses

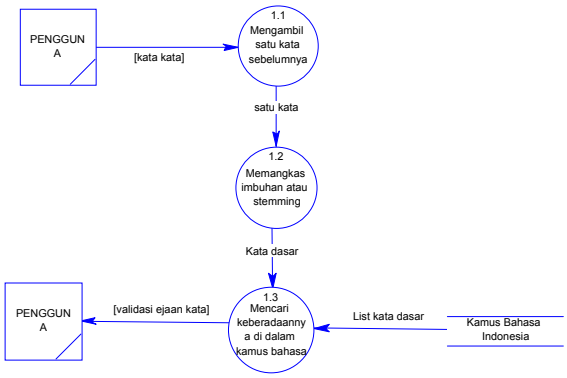
Setiap proses berhubungan dengan proses yang lain, oleh karena itu perlu adanya prioritas proses apa yang akan terlebih dahulu dilakukan. Pada Gambar 3.1 akan dijelaskan perancangan proses yang digunakan dalam penelitian ini.



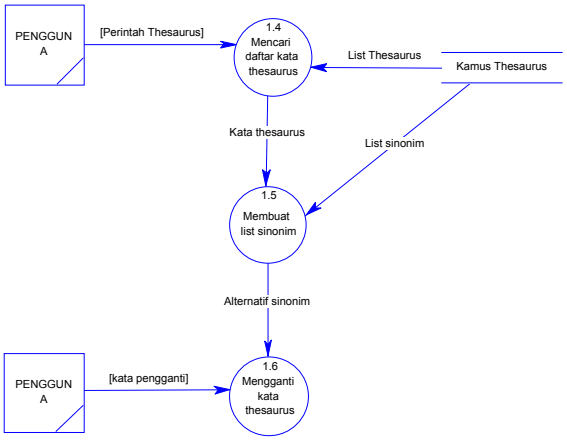
Gambar 3.1 Diagram Konteks DFD Level 0

Proses awal yang dilakukan adalah memasukkan kata-kata oleh pengguna aplikasi. Untuk setiap kata yang diketikkan akan dilakukan pengecekkan ejaan dengan menampilkan hasil warna merah untuk ejaan yang salah dan warna kata yang tidak berubah untuk ejaan yang benar.

Proses selanjutnya adalah pencarian sinonim thesaurus yang akan dilakukan setiap ada perintah pencarian oleh pengguna. Hasil seluruh sinonim yang dapat dicari oleh program akan ditampilkan untuk kemudian dipilih yang paling sesuai oleh pengguna. Untuk lebih jelasnya dapat dilihat pada Gambar 3.2 yang menampilkan DFD level 1 dari proses aplikasi pengolah kata.



Gambar 3.2 DFD Level 1 Proses Pengolah Kata



Gambar 3.3 DFD level 1 Proses Pengolah Kata(lanjut)

Dalam proses mengambil kata sebelumnya, digunakan pencacah kata yang akan mengambil satu kata dari kursor saat ini. Proses ini dipicu dari ketikan karakter pemisah kata, yaitu salah satu diantara karakter-karakter .<,>;:/?'\"|\\{}[]-_+=~!@#$%^&*(). Sebelum dilakukan pengecekkan kata tersebut di dalam kamus bahasa, maka dilakukan proses *stemming* terlebih dahulu. Informasi mengenai keberhasilan pengecekkan ejaan ini akan ditampilkan ke pengguna. Jika kata yang diketikkan valid, maka akan muncul informasi yang mengubah atribut kata tersebut. Jika tidak, maka atribut ejaan salah akan ditampilkan.

Proses selanjutnya yang mungkin terjadi adalah pengecekkan thesaurus yang dipicu oleh perintah pengguna. Setiapa ada perintah pengecekkan oleh pengguna, maka dilakukan proses pengambilan satu kata yang terletak sebelum kursor berada. Kemudian, program akan mencari kata tersebut di dalam kamus thesaurus yang telah dibangun sebelumnya. Jika terdapat di dalam kamus, program akan membuat list sinonim yang cocok dengan kata tersebut. List sinonim ini akan ditampilkan ke pengguna untuk kemudian dipilih mana yang paling cocok untuk menggantikan kata tersebut.

3.2 Perancangan Fitur

Perancangan Fitur meliputi beberapa hal sebagai berikut :

- Fitur Cari dan Ganti
  Pencarian dan penggantian teks yang ada di dalam aplikasi dapat dilakukan dengan memasukkan teks yang dicari.
- Fitur Pemilihan tipe dan ukuran Font
  Pemilihan Tipe dan ukuran Font dapat dilakukan dengan memilih daftar font yang tersedia di Windows yang ditampilkan dalam bentuk Combo Box.
- Fitur Potong, Salin dan Rekat
  Penyuntingan yang sering digunakan oleh Aplikasi-aplikasi pengolah kata lainnya adalah Potong, Salin dan Rekat yang terkait dengan teks yang disimpan di dalam clipboard. Sehingga dengan adanya fitur-fitur ini, hubungan antar Aplikasi Pengolah Kata yang terintegrasi dapat di implementasikan.
- Fitur Pengecekkan Ejaan
  Pengecekkan Ejaan dapat dilakukan untuk seluruh teks yang terdapat di dalam aplikasi. Untuk kata-kata yang tidak memenuhi ktriteria ejaan yang benar dalam bahasa Indonesia akan diberi warna yang berbeda dengan teks lainnya.
- Fitur PengecekkanThesaurus
  Pengecekkan Thesaurus dapat dilakukan untuk seluruh teks yang terdapat di dalam aplikasi. Untuk kata-kata yang tidak memenuhi ktriteria thesaurus yang benar dalam bahasa Indonesia akan diberi warna yang berbeda dengan teks lainnya.

3.3 Perancangan Masukan dan Keluaran

Teks yang akan digunakan sebagai masukan dapat diketikkan langsung ke dalam aplikasi atau berupa arsip yang bertipe Rich Text File (RTF) atau Text File sederhana (TXT). RTF dibuat oleh Microsoft dan merupakan format paling standar untuk menyunting teks yang dituliskan dalam aplikasi-aplikasi pengolah teks lainnya. Arsip masukan ini berisi beberapa text dengan mulit fonts dan format tertentu. Untuk membuka arsip masukan ini dijalankan dari menu Buka.

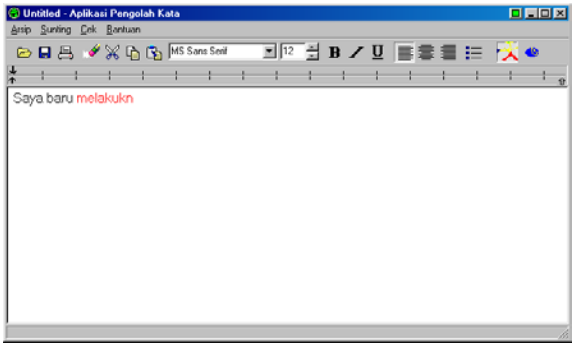
Teks yang dihasilkan dari aplikasi ini dapat disimpan ke dalam arsip yang bertipe Rich Text File (RTF) atau Text File sederhana (TXT). Arsip keluaran ini berisi beberapa text dengan mulit fonts dan format tertentu, tanpa adanya penyimpanan informasi hasil pengecekkan ejaan dan thesaurus. Untuk menyimpan arsip keluaran ini dijalankan dari menu Simpan atau Simpan Atas

4. UJI COBA PERANGKAT LUNAK

Sebelum perangkat lunak digunakan, terlebih dahulu dilakukan uji coba perangkat lunak untuk melihat kemampuan dari perangkat lunak tersebut.
Uji coba dilakukan dengan menggunakan komputer *server* dengan prosesor *Pentium III 750 MHz* dan memori 64 MB.

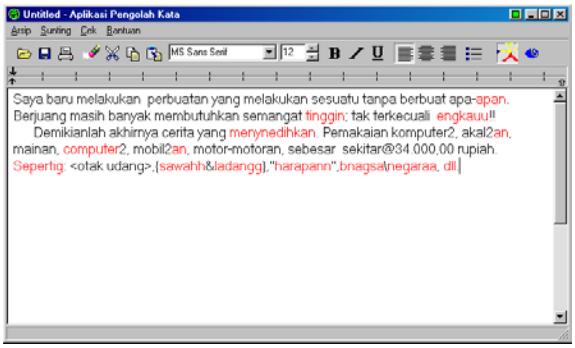
4.1 Uji Coba Pengecekkan Ejaan

Setiap pengguna aplikasi mengetikkan suatu kata yang diakhiri oleh pengetikkan karakter pembatas kata, yaitu: spasi(‘ ‘),titik(‘.’), koma(‘,’), titik koma(‘;’), titik dua(‘:’), dan karakter-karakter lainnya, akan dilakukan pengecekkan kata yang telah diketikkan sebelumnya. Kalau kata yang diketikkan sebelumnya terdapat didalam kamus bahasa Indonesia, maka warna kata tidak akan berubah. Dan apabila kata yang diketikkan sebelumnya tidak terdapat dalam kamus, maka warna kata akan berubah menjadi warna merah. Contoh pengetikkan kata yang salah dan benar dapat dilihat pada Gambar 4.1.



Gambar 4.1 Uji Coba Pengecekkan kata

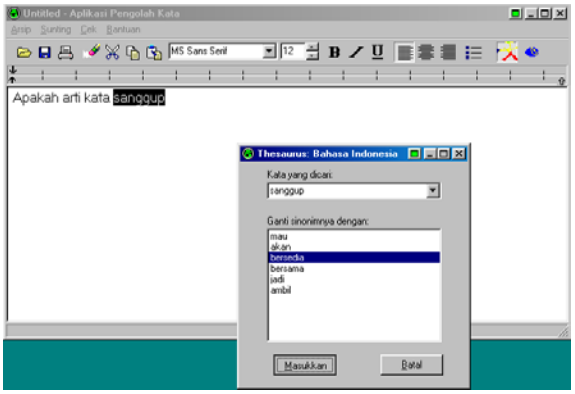
Dengan adanya pengecekkan ini, maka pengguna dapat langsung membetulkan kata yang salah, yaitu kata yang berwarna merah, sampai kata berwarna hitam kembali. Contoh pengujian seluruh kemungkinan yang terjadi di dalam pengecekkan kata dapat dilihat pada Gambar 4.2.



Gambar 4.2 Uji Coba Keseluruhan

4.2 Uji Coba Pengecekkan Thesaurus

Thesaurus akan dijalankan setelah pengguna aplikasi menekan tombol thesaurus atau menggunakan kunci shortcut (**SHIFT+F7**) pada akhir kata yang diinginkan dicari thesaurus-nya. Sebagai contoh dapat dilihat pada Gambar 4.3.



Gambar 4.3 Uji Coba Thesaurus

Maka pengguna dapat memilih kata yang dicari penggantinya dari daftar pilihan kata yang ada dengan menekan tombol 'Masukkan'. Maka secara otomatis kata sebelumnya akan diganti dengan pilihan pengguna.

5. KESIMPULAN dan SARAN

Setelah melakukan perancangan dan uji coba terhadap perangkat lunak yang dibuat, dapat diambil beberapa kesimpulan berikut:

1. Aplikasi ini diharapkan dapat membantu berbagai pihak (mahasiswa, penulis buku, wartawan, dan sebagainya) dalam mengolah kata menggunakan Bahasa Indonesia. Dengan adanya fasilitas pemeriksa ejaan, pengguna akan mendapatkan kemudahan dalam melakukan pengecekan terhadap kata-kata yang dituliskannya, sehingga mempersingkat waktu penyusunan tulisan karena pengguna tidak perlu lagi membaca satu persatu kata yang telah dituliskannya.
2. Sedangkan fasilitas yang lain, yaitu Thesaurus akan membantu seorang penulis dapat menemukan kata yang tepat dalam menyusun sebuah tulisan menggunakan Bahasa Indonesia.

Beberapa saran yang dapat dikemukakan untuk perbaikan performance dan manfaat perangkat lunak yang dibuat adalah sebagai berikut:

1. Digunakannya pengecekkan kata untuk arsip pengetikkan yang sudah jadi, sehingga arsip yang disimpan, setelah mengalami pengecekkan ejaan, tidak mempunyai atribut warna yang berwarna merah.
2. Pembangunan thesaurus hendaknya dibangun juga secara otomatis dalam modul program yang sama.
3. Digunakannya struktur data linking list dalam penyimpanan format thesaurus nya, sehingga seluruh sinonim dari kata yang dicari dapat di tampilkan secara rekursif.

5. DAFTAR PUSTAKA

[1] Robert R. Korfhage, Information Storage and Retrieval, Wiley Computer Publishing, Canada, 1997.
[2] G. Salton. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer, Addison-Wesley, Reading, Mass., 1989
[3] William B. Frakes dan Richardo B. Yates, Information Retrieval : Data Structures and Algorithms, Prentice-Hall, New Jersey, 1992
[4] Agus Zainal Arifin, Penggunaan Digital Tree Hibrida pada Aplikasi Information Retrieval untuk Dokumen Berita, Proseding Seminar Nasional Sains dan Teknologi 2002, Lembaga Penelitian, ITS, 31 Juli 2002.
[5] Alberto M., "Compound Key Word Generation from Document Databases Using A Hierarchical Clustering ART Model", Intelligent Data

Analysis, Vol. 1, No. 1, 1997 (http://www.elsevier.computing/locate/ida)

[6] Agus Zainal Arifin dan Ari Novan Setiono, Klasifikasi Dokumen Berita Kejadian Berbahasa Indonesia dengan Algoritma Single Pass Clustering, Proceeding of Seminar on Intelligent Technology and Its Applications (SITIA), Teknik Elektro, Institut Teknologi Sepuluh Nopember, 07 Mei 2002.

[7] Yiming Yang, dkk., "Learning Approaches for Detecting and Tracking News Events", IEEE Intelligent Systems, July/Agustus 1999, hal. 32-43.

[8] Mei Kobayashi, Koichi Takeda, " Information Retrieval on the Web: Selected Topic", Desember 1999.

[9] Yiming Yang, "An Evaluation of Statistical Approaches to Text Categorization", INRT Journal, 1998

[10] David D. Lewis, "Training Algorithms for Linear Text Classifiers", Proc. SIGIR '96: 19th Annual International ACM SIGIR Conference and Development in Information Retrieval, ACM Press, New York, Agustus 1996, hal. 298-306.

[11] Ames Allan, Ron Papka, dan Victor Lavrenko, "On-line New Event Detection and Tracking", Proc. SIGIR '98: 21st Annual International ACM SIGIR Conference and Development in Information Retrieval, Australia, Agustus 1998, hal. 37-45.