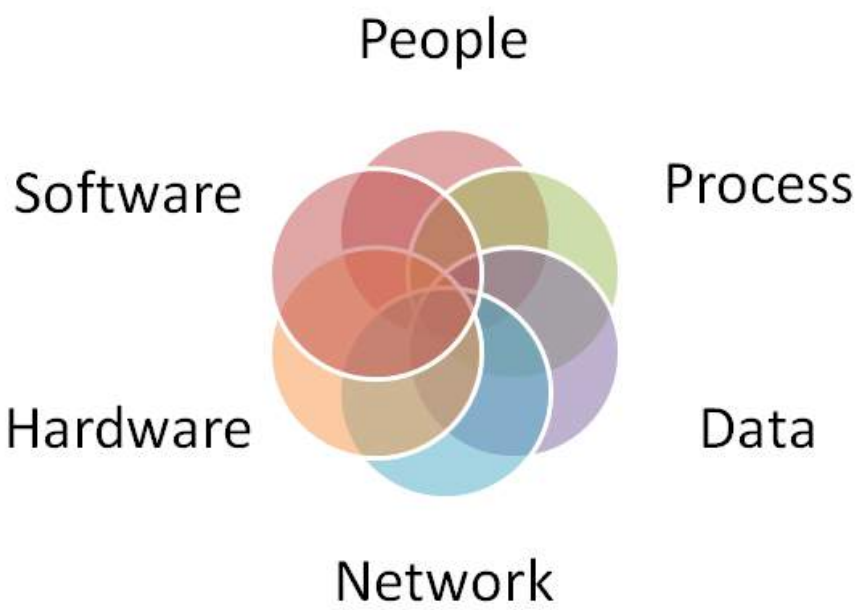
OAJIS
Open Access
Journal of
Information
Systems
is.its.ac.id/pubs/oajis/

ISSN 1979-3979

jurnal sisfo

Inspirasi Profesional Sistem Informasi



Vol. 07 No. 03 Mei 2018

OAJIS
Open Access
Journal of
Information
Systems
is.its.ac.id/pubs/oajis/

jurnal sisfo

Jurnal Sisfo Vol. 07 No. 03 (2018) i–ii

Sekretariat

Departemen Sistem Informasi – Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember (ITS) – Surabaya
Telp. 031-5999944 Fax. 031-5964965
Email: editor@jurnalsisfo.org
Website: http://jurnalsisfo.org

Jurnal SISFO juga dipublikasikan di *Open Access Journal of Information Systems* (OAJIS)

Website: http://is.its.ac.id/pubs/oajis/index.php

OAJIS
Open Access
Journal of
Information
Systems
is.its.ac.id/pubs/oajis/

jurnal sisfo

Jurnal Sisfo Vol. 07 No. 03 (2018) i–ii

Mitra Bestari

Ari Kusyanti, S.T., M.Sc. (Universitas Brawijaya)

Erma Suryani, S.T, M.T, Ph.D. (Institut Teknologi Sepuluh Nopember)

Dr. Eng. Febriliyan Samopa, S.Kom, M.Kom. (Institut Teknologi Sepuluh Nopember)

Nur Aini Rakhmawati, Ph.D (Institut Teknologi Sepuluh Nopember)

Dr. Ir. Rinaldi Munir, M.T. (Institut Teknologi Bandung)

Rahadian Bisma, M.Kom., ITILF. (Universitas Negeri Surabaya)

Renny Pradina Kusumawardani, S.T, M.T (Institut Teknologi Sepuluh Nopember)

Retno Aulia Vinarti, S.Kom, M.Kom. (Institut Teknologi Sepuluh Nopember)

Satria Fadil Persada, S.Kom., M.BA., Ph.D. (Institut Teknologi Sepuluh Nopember)

OAJIS
Open Access
Journal of
Information
Systems
is.its.ac.id/pubs/oajis/

jurnal sisfo

Jurnal Sisfo Vol. 07 No. 03 (2018) iii

Daftar Isi

Halaman ini sengaja dikosongkan

OAJIS
Open Access Journal of Information Systems
is.its.ac.id/pubs/oajis/

jurnal sisfo
Jurnal Sisfo Vol. 07 No. 03 (2018) 181–198

Memberdayakan Algoritma Knuth Morris Pratt untuk Pencarian dan Pemformatan Istilah Bahasa Inggris

Bonifacius Vicky Indriyono*

Sistem Informasi, Sekolah Tinggi Manajemen Informatika dan Komputer Kadiri

Abstract

The act of writing is one of the important things to be done by students who took the final project or thesis. In this activity, students will demonstrate their ability to ideas, ideas, understanding of the subject matter covered. Good writing is certainly expected to be understood by the reader. Not infrequently in the text contains a lot of foreign terms, especially English. In general, in order to clarify its use, the term in English was printed miring.Namun there's also writing an English term that has not been in italics. The main reason is the difficulty in finding and reformat the term. To facilitate the search term English language, it can be used algorithm Knuth Morris Pratt (KMP). This research resulted in an application using the KMP algorithm. These algorithms will match the English term that is in the vocabulary database of English with English terms that are in the document. If there is a match, then the term will be automatically italics.

Keywords: String, Exact Matching, Algorithm, KMP

Abstrak

Kegiatan menulis merupakan salah satu hal penting yang harus dilakukan oleh mahasiswa yang menempuh tugas akhir maupun skripsi. Dalam kegiatan ini, mahasiswa akan menunjukkan kemampuannya dalam menuangkan ide, gagasan, pikiran, pemahaman tentang topik permasalahan yang dibahas. Tulisan yang baik tentunya diharapkan mampu dipahami dan dimengerti oleh pembaca. Tidak jarang dalam tulisan tersebut mengandung banyak istilah asing khususnya bahasa Inggris. Pada umumnya, untuk memperjelas penggunaannya, istilah dalam bahasa Inggris tersebut dicetak miring.Namun masih banyak dijumpai penulisan istilah bahasa Inggris yang belum tercetak miring. Alasan utamanya adalah kesulitan dalam mencari dan menformat istilah tersebut. Untuk mempermudah pencarian istilah bahasa Inggris tersebut, maka dapat digunakan algoritma Knuth Morris Pratt (KMP). Penelitian yang dilakukan ini menghasilkan sebuah aplikasi yang memanfaatkan algoritma KMP. Algoritma ini akan mencocokan istilah bahasa Inggris yang berada di *database* kosakata bahasa Inggris dengan istilah bahasa Inggris yang berada di dokumen. Apabila terdapat kecocokan, maka istilah tersebut akan otomatis tercetak miring.

Kata kunci: String*, Exact Matching*, Algoritma, KMP

Histori Artikel : Disubmit 9 April 2018; Diterima 4 Mei 2018; Tersedia online 7 Mei 2018

*Corresponding Author
Email address: bonifaciusvicky@gmail.com (Bonifacius Vicky Indriyono)

1. Pendahuluan

Kegiatan menulis merupakan salah satu hal penting yang dilakukan oleh mahasiswa yang menempuh tugas akhir maupun skripsi. Dalam kegiatan penulisan ini, mahasiswa akan menunjukkan kemampuannya dalam menuangkan ide, gagasan, pikiran, pemahaman tentang topik permasalahan dan seni dalam menulis, sehingga apa yang ditampilkan dalam tulisan tersebut diharapkan mudah dibaca, mampu membuat pembaca merasa enak, paham dan dapat mengerti apa yang dipaparkan.Tidak jarang dalam kegiatan penulisan tugas akhir maupun skripsi tersebut banyak menggunakan istilah-istilah asing khususnya istilah bahasa Inggris. Pada umumnya, untuk memperjelas penggunaannya, maka istilah-istilah dalam bahasa Inggris tersebut dicetak miring (*italic*). Namun pada kenyataannya, masih banyak dijumpai penulisan istilah bahasa Inggris yang belum tercetak miring dalam tugas akhir maupun skripsi mahasiswa. Sering kali dijumpai kesalahan penulisan istilah bahasa Inggris ini, sehingga pada waktu mahasiswa melakukan bimbingan dengan dosen pembimbing, dosen akan memberikan perbaikan terhadap penulisan istilah bahasa Inggris tersebut. Jika dalam naskah ternyata banyak terdapat istilah bahasa Inggris dan tidak semuanya dicetak miring, maka akan sangat membutuhkan waktu yang lama untuk mencari dan mencetak miring seluruh istilah asing tersebut. Kebutuhan waktu yang lama disebabkan karena harus dilakukan pencarian satu per satu istilah-istilah asing tersebut untuk selanjutnya satu per satu di cetak miring. Maka untuk mengatasi masalah tersebut, perlu dibuat sebuat perangkat lunak yang dapat mendeteksi istilah bahasa Inggris dan secara otomatis mencetak miring istilah yang ditemukan tersebut.

Penggunaan fungsi pencarian *string* dalam kegiatan pembuatan perangkat lunak sudah menjadi hal umum yang biasa digunakan. *String matching* yang dalam bahasa Indonesia dikenal dengan istilah pencocokan string merupakan suatu algoritma yang digunakan untuk melakukan pencarian semua kemunculan *string* pendek *pattern [0……n-1]* yang disebut *pattern* di *string* yang lebih panjang teks *[0……m-1]* yang disebut teks [1]. Beberapa algoritma dapat diterapkan untuk proses *string matching* ini, salah satunya adalah algoritma Knuth Morris Pratt (KMP). Algoritma KMP merupakan jenis *Exact String Matching Algorithm* yang merupakan pencocokan *string* secara tepat dengan susunan karakter dalam *string* yang dicocokkan memiliki jumlah maupun urutan karakter dalam *string* yang sama [2]. Sebagai contoh sederhana adalah *string* ***entity*** hanya akan menunjukkan kecocokan dengan *string* ***entity***.

Dalam penelitian ini, dihasilkan sebuah perangkat lunak aplikasi yang dapat digunakan untuk melakukan pencarian *string* istilah bahasa Inggris. Algoritma yang diterapkan dalam aplikasi adalah algoritma Knuth Morris Pratt (KMP). Prosedur yang dijalankan dalam aplikasi ini adalah program akan melakukan pencarian *string* kemudian membandingkan antara *string* istilah bahasa Inggris yang terdapat dalam *database* kosakata bahasa Inggris dengan *string* istilah bahasa Inggris yang terdapat dalam naskah. Apabila pada saat pencarian dan pencocokan ditemukan *string* istilah bahasa Inggris yang cocok, maka selanjutnya istilah tersebut akan tercetak *italic* (cetak miring) secara otomatis.

Masalah pemformatan istilah yang termasuk dalam kosakata bahasa Inggris adalah menjadi salah satu hal yang wajib diperhatikan oleh mahasiswa yang sedang dalam proses penyusunan tugas akhir maupun skripsi. Hal ini penting untuk diperhatikan dan dilakukan karena format *italic* (cetak miring) untuk setiap istilah dalam bahasa Inggris menjadi salah satu syarat penulisan artikel ilmiah yang baik. Dari uraian permasalahan diatas, maka dapat diambil rumusan masalah sebagai berikut: 1) Bagaimanakah penerapan algoritma Knuth-Morris-Pratt (KMP) dalam sebuah aplikasi untuk pencarian dan pencocokan suatu string istilah bahasa Inggris?, 2) Bagaimana tingkat keberhasilan penerapan algoritma KMP dalam proses pencarian dan pemformatan string istilah bahasa Inggris dalam penelitian ini?

Agar pembahasan dalam penelitian ini lebih terarah, maka diberikan beberapa batasan masalah sebagai berikut: 1) Istilah bahasa Inggris yang digunakan adalah kosakata bahasa Inggris umum dan kosakata bahasa Inggris komputer yang sering digunakan sehari-hari, 2) *Database* kosakata bahasa Inggris dibuat dengan menggunakan DBMS MySQL, 3) Naskah yang digunakan untuk proses pencarian *String* dalam format *text file* dengan ekstensi.doc dan .docx, 4) Penelitian dilakukan di Sekolah Tinggi Manajemen Informatika dan

Komputer Kadiri (STMIKKA) Kediri berdasarkan pada file naskah tugas akhir maupun skripsi mahasiswa, 5) Aplikasi yang dibangun untuk mengimplementasikan hasil analisis teknik *String Matching* dan algoritma KMP adalah dengan menggunakan *compiler* Delphi 2010.

Beberapa tujuan yang hendak dicapai dalam penelitian ini adalah sebagai berikut: 1) Mengetahui konsep algoritma pencarian Knuth-Morris-Pratt (KMP), 2) Dapat menerapkan konsep algoritma KMP untuk mencari *string*/istilah bahasa Inggris yang terdapat dalam sebuah naskah 3) Dapat mengimplementasikan algoritma KMP dalam sebuah perangkat lunak pencari dan pemformat istilah bahasa Inggris.

2. Tinjauan Pustaka/Penelitian Sebelumnya

Bagian tinjauan pustaka/penelitian sebelumnya digunakan untuk mempelajari dan melihat hasil dari beberapa penelitian yang dilakukan oleh peneliti terdahulu yang relevan dengan topik penelitian yang dilakukan peneliti sekarang. Beberapa penelitan tentang pemanfaaatan *String Matching* dan algoritma KMP telah dilakukan oleh peneliti sebelumnya diantaranya:

1) Sri Lestari, Amin Djaya [3] dalam prosiding yang berjudul "Aplikasi Search Engine Menggunakan Algoritma Knuth Morris Pratt", menjelaskan tentang cara pencarian berkas dalam komputer menggunakan algoritma KMP. Kesamaan dengan penelitian yang dilakukan penulis sekarang adalah sama-sama memanfaatkan *database* untuk menampung data/berkas dan menggunakan algoritma KMP untuk melakukan pencarian berkas. Perbedaannya terletak pada obyek yang dicari. Pada penelitian sebelumnya, pencarian dilakukan terhadap *file*/*folder*, sedangkan pada penelitian ini, obyek yang dicari adalah *string* istilah bahasa Inggris yang terdapat dalam sebuah naskah bertipe .doc/.docx.
2) Thio Wibowo, Ardianto Wibowo, Rika Perdana Sari [4] dalam jurnal yang berjudul "Pembuatan Aplikasi Untuk Mendeteksi Kebenaran Perintah Sql Query Menggunakan Metode Knuth-Morris Pratt (KMP)", menjelaskan tentang cara pencocokan *query* dalam *database* dengan *query* yang diinputkan mahasiswa. Penelitian sebelumnya dengan penelitian yang dilakukan sekarang memiliki kesamaan dalam hal pencarian dan pencocokan *string* dengan algoritma KMP. Perbedaan terletak hasil akhir. Jika pada penelitian sebelumnya hasil akhir berupa pembobotan pada hasil pencarian *query*, sedangkan pada penelitian yang dilakukan sekarang hasil akhir *string* yang ditemukan dan cocok dengan basis data akan tercetak miring secara otomatis.
3) Ibrahim M. Abu-Zaid, Emad Kh. El-Rayyes [5] dalam jurnal yang berjudul "*Parallel Search Using KMP Algorithm in Arabic String*" menjelaskan tentang teknik pararel pencarian dan pencocokan teks bahasa Arab. Persamaan dengan penelitian sekarang adalah proses yang sama dalam mencari dan mencocokkan teks dengan algoritma KMP. Perbedaan terletak pada obyek yang dicari. Pada penelitian sebelumnya mencari teks bahasa Arab, sedangkan pada penelitian sekarang obyek yang di cari adalah istilah-istilah dalam bahasa Inggris.
4) Halima Tus Sa'diah [6] dalam jurnal yang berjudul "Implementasi Algoritma *Knuth-Morris-Pratt* Pada Fungsi Pencarian Judul Tugas Akhir *Repository*" menjelaskan tentang penerapan algoritma KMP untuk sistem pencarian pada repository tugas akhir. Perbedaan dengan penelitian yang dilakukan sekarang adalah pada penerapan algoritma Knuth Morris Pratt. Penelitian sebelumnya, menerapkan algoritma Knuth Morris Pratt dalam sebuah aplikasi untuk menemukan tugas akhir pada sebuah repository, sedangkan pada penelitian sekarang diterapkan dalam sebuah aplikasi *search engine* yang berbasis *desktop*.
5) Fince Tinus Waruwu dan Rila Mandala [7] dalam jurnal yang berjudul "Perbandingan Algoritma Knuth Morris Pratt Dan Boyer Moore Dalam Pencocokan String Pada Aplikasi Kamus Bahasa Nias." menjelaskan tentang perbandingan penerapan algoritma KMP dan Boyer Moore untuk pencarian teks dalam kamus bahasa Nias. Perbedaan utama dengan penelitian yang dilakukan sekarang adalah pada hasil implementasi algoritma KMP. Pada penelitian terdahulu implementasi algoritma digunakan untuk menemukan tingkat efisiensi penggunaan algoritma pencarian *string* antara KMP dengan Boyer Moore,

sedangkan pada penelitian sekarang hasil implementasi KMP diterapkan untuk mencari istilah bahasa Inggris dalam sebuah aplikasi berbasis *desktop*.

6) Maya Rossaria, Boko Susilo, Ernawati [8] dalam jurnal berjudul Implementasi Algoritma Pencocokan *String Knuth Morris Pratt* Dalam Aplikasi Pencarian Dokumen *Digital* Berbasis Android, memaparkan tentang bagaimana algoritma pencocokan *string* Knuth Morris Pratt dimanfaatkan untuk menemukan dokumen digital yang terdapat pada Android. Perbedaan dengan penelitian sekarang adalah terletak pada implementasi dari algoritma KMP. Pada penelitian sebelumnya, algoritma KMP digunakan untuk melakukan pencarian berkas pada sistem operasi android, sedangkan penelitian saat ini algoritma KMP dimanfaatkan untuk melakukan deteksi dan pencocokan *string* bahasa Inggris dalam sebuah naskah dengan menggunakan aplikasi berbasis *desktop*.

Berikut ini adalah beberapa pustaka yang digunakan untuk menunjang pelaksanaan kegiatan penelitian ini.

2.1 Pengertian String

String dapat di definisikan sebagai suatu susunan dari karakter-karakter yang terdiri dari angka, alphabet, atau karakter yang. *String* dapat berupa kata, frase, atau kalimat lain yang biasanya direpresentasikan sebagai struktur data array. Menurut Kadir [9], *string* adalah untaian karakter dengan panjang tertentu yang dapat diperlakukan sebagai tipe dasar dalam pemrograman yang disusun dari elemen-elemen bertipe karakter. *String* dan karakter memiliki perbedaan yang terletak padacara penulisannya. *String* ditulis dengan diapit oleh tanda petik ganda (“teks“), sedangkan karakter ditulis dengan diapit oleh tanda petik tunggal (‘teks‘) .

2.2 Pengertian String Matching

Algoritma *string matching* dalam bahasa Indonesia dikenal dengan istilah algoritma pencocokan *string* [10]. *String matching* adalah pencarian sebuah *pattern* pada sebuah teks [11]. Menurut (Riyanarto Sarno, Yeni Anistyasari, dan Rahimi Fitri) [12], yang dimaksud dengan *string matching* adalah proses pencarian semua kemunculan *query* yang selanjutnya disebut *pattern* ke dalam *string* yang lebih panjang (teks). *Pattern* dilambangkan dengan $x=x[0..m-1]$ dan panjangnya adalah m. Teks dilambangkan dengan $y=y[0..n-1]$ dan panjangnya adalah n. *String matching* sendiri dibagi menjadi dua, yaitu *exact matching* dan *heuristic matching*. Algoritma *string matching* adalah sebuah algoritma yang digunakan dalam pencocokkan suatu pola kata tertentu terhadap suatu kalimat atau teks panjang. Algoritma *string matching* sendiri dapat dilakukan dengan beberapa cara tertentu, antara lain cara *Brute Force* dan cara *KnuthMorris-Pratt* (KMP) [13].

2.3 Pengertian Exact String Matching

Exact string matching didefinisikan sebagai proses pencocokan *string* secara tepat dimana susunan karakter dalam *string* yang dicocokkan memiliki jumlah maupun urutan karakter dalam *string* yang sama [14]. *Exact string matching* ini digunakan untuk menemukan *pattern* yang berasal dari suatu teks yang dimasukkan [12]. Sebagai contoh terdapat teks “**Saya adalah seorang guru”** dan terdapat *pattern* **guru.** Kalimat pertama tertulis “**Saya adalah guru di SMA ABC”** sedangkan kalimat kedua tertulis **“Profesi saya adalah seorang pendidik”**. Pada saat dilakukan pencarian secara *exact* maka sistem akan memberikan hasil bahwa yang memiliki kecocokan adalah kalimat pertama karena mengandung kata **guru** sedangkan pada kalimat kedua tidak ditemukan kecocokan, meskipun sebenarnya kata **guru** dan **pendidik** adalah kata yang bersinonim (memiliki makna yang sama).

2.4 Prinsip Kerja String Matching

Menurut (Effendi) [15], prinsip kerja dari algoritma *string matching* meliputi: 1) Memindai teks dengan bantuan sebuah *window* yang ukurannya sama dengan panjang *pattern,* 2) Menempatkan *window* pada awal teks, 3) Membandingkan karakter pada *window* dengan karakter dari *pattern*. Setelah pencocokan (baik hasilnya cocok atau tidak cocok) dilakukan pergeseran ke kanan pada *window*. Prosedur ini dilakukan

berulang-ulang sampai *window* berada pada akhir teks. Mekanisme ini disebut mekanisme *sliding window.* Masih menurut Effendi [15], algoritma *string matching* memiliki 3 buah komponen utama yakni : **Pattern** : adalah deretan karakter yang akan dicocokkan dengan teks, dinyatakan dengan $x[0 \ldots m-1]$, panjang *pattern* dinyatakan dengan m. Gambar 1 di bawah ini memperlihatkan contoh dari cara kerja *string matching*, **Teks** yang didefinisikan sebagai tempat pencocokan *pattern* dilakukan. Dinyatakan dengan $y[0 \ldots n-1]$, panjang teks dinyatakan dengan n dan **Alfabet** yang berisi semua simbol yang digunakan oleh bahasa pada teks dan *pattern*, dinyatakan dengan $\sum$ dengan ukuran dinyatakan ASIZE.

Gambar 1 Contoh hasil cara kerja *string matching*

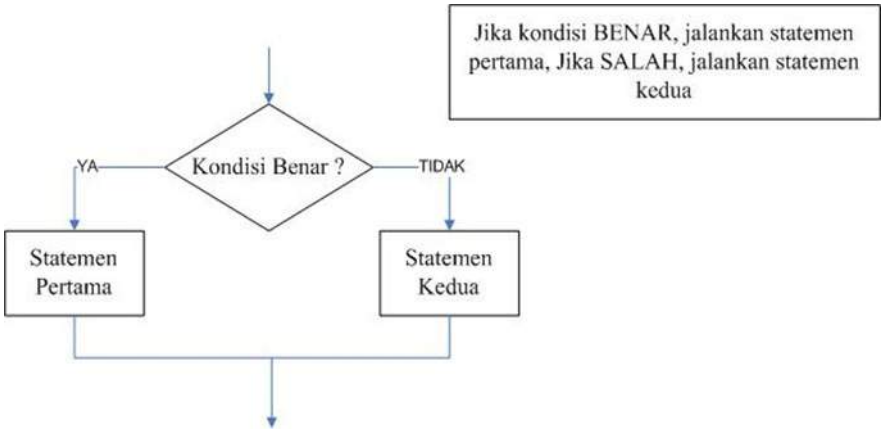
2.5 Pengertian Algoritma

Menurut Munir [16], yang dimaksud dengan algoritma adalah urutan langkah-langkah logis dalam menyelesaikan masalah yang disusun secara sistematis. Langkah-langkah dalam algoritma harus logis dan harus dapat ditentukan apakah bernilai salah atau benar, sedangkan menurut Sjukani [17], algoritma merupakan suatu alur pikiran dalam menyelesaika suatu pekerjaan yang dituangkan dalam bentuk tertulis yang dapat dimengerti oleh orang lain, tersusun secara logis, dan dapat diselesaikan dengan benar. Dalam penulisan algoritma ini pada umumnya menggunakan bahasa natural atau menggunakan notasi matematika, sehingga masih belum dapat dijalankan pada komputer. Penulisan algoritma ini akan lebih baik jika disusun secara sistematis menggunakan beberapa skema diantaranya *flowchart, psedudoce* dan sebagainya. Langkah-langkah logis algoritma dalam menyelesaikan sebuah masalah tersebut dapat berupa runtunan aksi (*sequence*), pemilihan aksi (*selection*), pengulangan aksi (*iteration*) atau kombinasi dari ketiganya. Berikut ini penjelasan dari ketiga langkah logis dalam algoritma:

1) **Struktur Runtunan:** Digunakan untuk program yang pernyataannya *sequential* atau urutan. Ilustrasi dari struktur runtunan ini diperlihatkan pada Gambar 2.

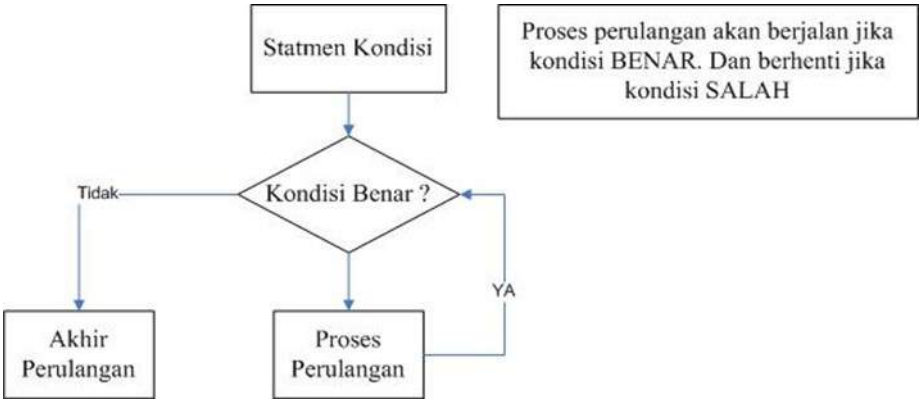
Gambar 2 Ilustrasi struktur runtunan

2) **Struktur Pemilihan:** Digunakan untuk program yang menggunakan pemilihan atau penyeleksian kondisi. Ilustrasi struktur pemilihan diperlihatkan pada Gambar 3.



Gambar 3 Ilustrasi struktur pemilihan

3) **Struktur Perulangan:** Digunakan untuk program yang pernyataannya akan dieksekusi berulang-ulang. Ilustrasi dari struktur perulangan diperlihatkan pada Gambar 4.



Gambar 4 Ilustrasi struktur perulangan

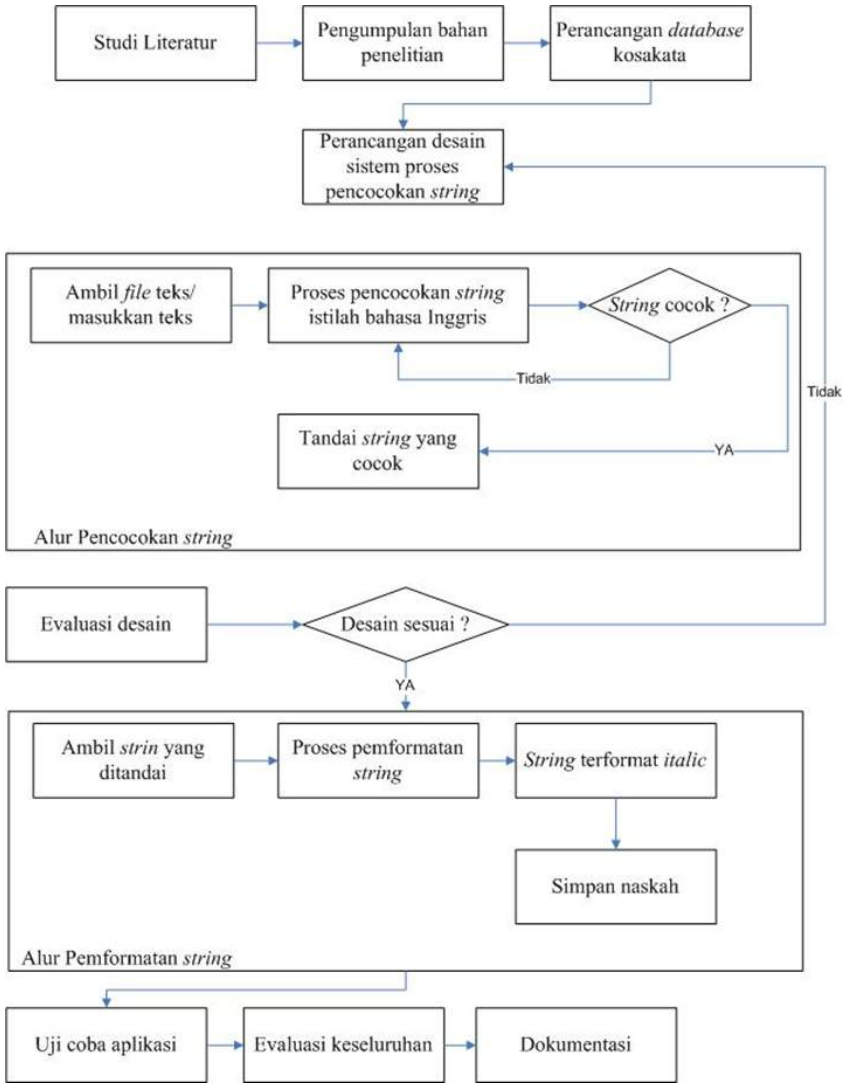
2.6 Pengertian Algoritma Knuth Morris Pratt

Algoritma Knuth Morris Pratt (KMP) merupakan salah satu algoritma yang digunakan untuk melakukan proses pencocokan *string*. Pada algoritma KMP, kita memelihara in*Form*asi yang digunakan untuk melakukan jumlah pergeseran. Algoritma menggunakan in*Form*asi tersebut untuk membuat pergeseran yang lebih jauh, tidak hanya satu karakter [18]. Contoh kata ***entity*** akan menunjukkan kecocokan hanya dengan kata ***entity***. Algoritma Knuth Morris Pratt digunakan untuk memproses dua karakter (atau *byte*) secara paralel, untuk mempercepat proses pencocokan *pattern* [19]. Pada waktu proses pencocokan berlangsung, apabila terjadi ketidakcocokan pada saat *pattern* sejajar dengan teks *[i..i + n -1],* maka hal itu dapat dianggap sebagai ketidakcocokan pertama yang terjadi di antara teks *[i + j]* dan *pattern [j],* dengan *<j* [12]. Secara sistematis, langkah-langkah yang dilakukan algoritma Knuth-Morris-Pratt pada saat mencocokkan *string* adalah sebagai berikut [13]:

1) Masukkan *query* kata yang akan dicari. Dengan permisalan P adalah sebuah *Pattern* atau pola susunan kata yang dijadikan sebagai contoh atau pola teks yang akan dicari dan T adalah sebuah Teks atau judul dokumen

2) Algoritma Knuth-Morris-Pratt mulai mencocokkan *pattern* atau pola susunan kata yang dijadikan sebagai contoh pada awal teks.
3) Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per karakter *pattern* atau pola susunan kata yang dijadikan sebagai contoh dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
   a. Karakter di *pattern* atau pola susunan kata yang dijadikan sebagai contoh dan di teks yang dibandingkan tidak cocok (*mismatch*).
   b. Semua karakter di *pattern* atau pola susunan kata yang dijadikan sebagai contoh cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
4) Algoritma kemudian menggeser *pattern* atau pola susunan kata yang dijadikan sebagai contoh, lalu mengulangi langkah No. 2 sampai *pattern* atau pola susunan kata yang dijadikan sebagai contoh berada di ujung teks.

3. Metodologi



Gambar 5 Alur Penelitian

3.1 Metode Penelitian

Metode penelitian yang digunakan oleh penulis dalam penelitian ini adalah metode deskriptif. Dalam metode ini beberapa langkah yang dilakukan oleh peneliti diantaranya mengumpulkan bahan-bahan penelitian, mencari sumber pustaka dan literatur yang terkait dengan penelitian yang dibahas, mempelajari jalannya algoritma Knuth Morris Pratt (KMP), melakukan perancangan sistem, melakukan desain perangkat lunak yang akan dihasilkan dengan berdasarkan pada aturan algoritma KMP dan melakukan uji coba terhadap perangkat lunak. Secara umum, langkah-langkah implementasi yang dilakukan dalam penelitian ini adalah sebagai berikut: 1) Merancang *database* kamus istilah bahasa Inggris komputer, 2) Memasukkan teks yang akan dicocokkan dengan basis data, 3) Melakukan proses pencocokan *string*, 4) Menandai *string* yang cocok, 5) Melakukan pemformatan *italic* terhadap istilah yang ditemukan kecocokannya.

3.2 Kerangka Kerja/Alur Penelitian

Kerangka kerja/alur penelitian adalah tahapan proses yang dilakukan oleh peneliti dalam mengkaji dan membangun aplikasi pencarian dan pemformatan *string* istilah bahasa Inggris dengan algoritma KMP. Tahapan dalam penelitian ini diperlihatkan seperti pada Gambar 5. Berdasarkan pada Gambar 5, maka dapat diuraikan langkah-langkah dalam alur penelitian sebagai berikut:

1) **Studi literatur**. Studi literatur ini dilakukan untuk mencapai tujuan yang telah ditentukan. Selanjutnya, literatur-literatur yang dikumpulkan diseleksi. Proses ini dilakukan untuk menentukan literatur-literatur yang akan dipelajari dan digunakan dalam penelitian. Melalui studi literatur ini, peneliti dapat mempelajari teori-teori yang berhubungan dengan pengenalan pola *(pattern),* dan algoritma Knuth Morris Pratt. Teori-teori yang dipelajari tersebut diambil berdasarkan pada sumber baik berupa jurnal ilmiah, prosiding, buku dari situs internet.
2) **Pengumpulan bahan penelitian**. Pada tahap ini, peneliti mulai mengumpulkan bahan-bahan yang digunakan untuk menunjang penelitian. Bahan-bahan penelitian yang dimaksud berupa daftar kosakata istilah bahasa Inggris yng biasa dipergunakan sehari-hari dan contoh dokumen yang berasal dari laporan mahasiswa STMIK Kadiri, baik itu berupa laporan kerja praktek (KP), tugas akhir (untuk D3) dan laporan skripsi (untuk S1). Dokumen dari laporan-laporan tersebut itu nantinya yang akan digunakan sebagai bahan uji coba pencarian *string* bahasa Inggris.
3) **Perancangan database kosakata**. Perancangan database kosakata ini dilakukan oleh peneliti untuk menampung semua daftar kosakata istilah bahasa Inggris yang akan dijadikan sebagai *pattern* dalam proses pencarian dan pencocokan *string*.
4) **Perancangan desain sistem proses pencocokan string**. Pada langkah ini, peneliti mulai merancang desain antar muka untuk melakukan proses pencocokan *string*. Desain antar muka dan kode perintah disesuaikan dengan alur dari algoritma KMP yang telah dipelajari dan dianalisis sebelumnya.
5) **Evaluasi desain**. Proses ini dilakukan untuk mengetahui apakah desain antar muka telah sesuai dengan prinsip kerja algoritma KMP atau belum. Apabila desain belum sesuai, maka dilakukan re-desain terhadap antar muka tersebut.
6) **Uji coba aplikasi**. Setelah proses evaluasi desain selesai dilakukan dan hasil evaluasi diperbaiki, maka langkah selanjutnya adalah melakukan uji coba aplikasi. Uji coba dilakukan untuk melihat dan mengetahui sejauh mana tingkat ketepatan jalannya aplikasi berdasarkan pada desain sistem yang telah dibuat, mulai dari tahap memasukkan teks, melakukan proses pencocokan *string* dan sampai tahap pemformatan *string* yang ditemukan menjadi *italic* (cetak miring).
7) **Evaluasi keseluruhan**. Proses ini dilakukan setelah melakukan tahap uji coba. Evaluasi keseluruhan dimaksudkan untuk mengetahui keseluruhan jalannya sistem yang diterapkan dalam aplikasi apakah sudah sesuai dengan prinsip kerja dari algoritma KMP dan mencatat tiap kekurangan yang ada dalam hasil implementasi. Hasil yang didapatkan dari evaluasi ini adalah bahwa aplikasi dapat menerapkan prinsip dari algoritma KMP yaitu mencari dan menemukan *string* yang susunannya sama tepat dengan *pattern*. Kekurangan yang didapatkan adalah, apabila ada kesalahan pengetikkan *string* yang secara kasat mata merupakan istilah bahasa Inggris, maka aplikasi tidak dapat mendeteksi.

8) **Dokumentasi**. Tahap dokumentasi merupakan tahap akhir dalam penelitian yang dilakukan. Dalam tahap ini, seluruh rancangan sistem, alur jalannya aplikasi didokumentasikan dan disimpan dalam media penyimpanan. Proses penelitian di dokumentasikan dalam sebuah makalah penelitian.

3.3 Metode Analisis Dengan Fungsi Pinggiran

Fungsi pinggiran di definisikan sebagai fungsi yang digunakan untuk mengindikasikan pergeseran *pattern* P terbesar yang mungkin dengan menggunakan perbandingan yang dibentuk sebelum pencarian *string* [20]. Masih menurut Munir [20], fungsi pinggiran dihitung hanya berdasarkan kepada karakter-karakter dalam *pattern*, tidak menyertakan karakter-karakter dalam teks (*string* target). Fungsi pinggiran *b(j)* didefinisikan sebagai ukuran awalan terpanjang dari *pattern P* yang merupakan akhiran dari *P[1..j]*. Berikut ini diberikan conoh penerapan perhitungan fungsi pinggiran. Diketahui sebuah teks *T*= **MONIKRENI** dan *pattern P*=**MONIK**, maka tabel fungsi pinggirannya diperlihatkan dalam Tabel 1 berikut ini :

Tabel 1. Contoh tabel fungsi pinggiran

J	0	1	2	3	4	5
P[j]	M	O	N	I	K	R
B(j)	0	0	0	1	2	0

Dari tabel 1 diatas, sekarang mulai kita melakukan percobaan mencocokkan *pattern* P tadi dengan teks *T* = MONIKRENI.

I	0	1	2	3	4	5	6	7	8
T	M	O	N	I	K	R	E	N	I

J	0	1	2	3	4	5
P	M	O	N	I	K	A

Untuk menyelesaikan perhitungan, maka langkah pertama adalah membandingkan ujung kiri *pattern P* dengan ujung kiri teks-teks *T*. Karakter-karakter yang terdapat pada posisi 0 sampai dengan 4 memiliki kesamaan (*match*), tetapi pada posisi *i = j = 5* karakternya tidak cocok, karakter *R* pada teks *T* dengan karakter *A* pada *pattern P*. Karena terjadi ketidakcocokan, maka dapat dilakukan lagi pergeseran *pattern P*. Jumlah pergeserannya sesuai dengan nilai pinggiran dari awalan *pattern P* yang memiliki kesamaan. Pada kasus diatas, awalan yang sesuai adalah MONIK dengan panjang *(l) = 5*. Nilai pinggiran yang terpanjang untuk string *P[0..4]* adalah *b(4) = 2*. Jumlah pergeseran adalah : ***l – b*** sehingga ***5 – 2 = 3***. Jadi hasil akhirnya, *pattern P* digeser sejauh 3 karakter ke kanan.

3.4 Metode Analisis Dengan Algoritma Knuth Morris Pratt

Dalam penelitian ini, data yang didapatkan akan dianalisis dengan algoritma Knuth Morris Pratt. Terdapat dua (2) buah hal utama yang biasa ada dalam proses pencarian *string* ini, diantaranya : 1) **Teks**: teks adalah *string* yang relatif panjang (kumpulan dari karakter) yang akan menjadi bahan yang akan dicari kecocokannya dengan sebuah *pattern* (misalnya panjang dari sebuah teks dianalogikan dengan simbol: *X*), 2) ***Pattern***: adalah *string* yang panjangnya relatif lebih pendek dari teks (misalkan panjang *pattern* dilambangkan dengan *Y*, sehingga *Y<X*). *Pattern* ini nanti yang akan digunakan sebagai *string* yang dicocokkan ke dalam teks. Algoritma KMP melakukan proses awal (*preprocessing*) terhadap *pattern P* dengan menghitung fungsi pinggiran.

Berikut ini diberikan contoh analisis data dengan algoritma KMP. Terdapat sebuah teks yang berjudul **STMIKKA**. Dari teks tersebut diberikan *pattern* yang akan dicari dan dicocokkan adalah **MIKKA**. Dari contoh tersebut maka dapat diberikan cara kerja penyelesaiannya sebagai berikut:

1) Terlebih dahulu menentukan *pattern* dan teks yang akan dicari. Dalam kasus ini, *pattern* dan teks yang akan dicari adalah sebagai berikut: *Pattern P* = **MIKKA** dan teks *T* = **STMIKKA**
2) Berikutnya adalah menentukan fungsi pinggiran dari *pattern* dan teks yang akan dicari. Fungsi pinggiran didefinisikan sebagai ukuran awalan terpanjang dari *pattern.* Fungsi pinggiran tersebut dapat dilihat dalam Tabel 2 berikut ini:

Tabel 2. Tabel fungsi pinggiran kasus pencarian

J	0	1	2	3	4	5	6
P[j]	S	T	M	I	K	K	A
B(j)	0	1	3	0	0	0	0

3) Membuat peng-*index*-an atau memberikan nilai indeks terhadap *pattern* dan teks. Adapun nilai indeks *pattern* dan teks diperlihatkan dalam Tabel 3 dan Tabel 4 berikut.

Tabel 3. Tabel nilai indeks untuk teks

i	0	1	2	3	4	5	6
t[j]	S	T	M	I	K	K	A

Tabel 4. Tabel nilai indeks untuk *pattern*

i	0	1	2	3	4
p[j]	M	I	K	K	A

4) Dengan menggunakan algoritma KMP, kita lakukan proses perhitungan sebagai berikut :
   a. Membandingkan ujung kiri teks dan ujung kiri pada *Pattern.* Pada ujung kiri teks dan ujung kiri *pattern* terjadi ketidakcocokan yaitu teks dengan karakter pada indeks pertama *S* dan karakter pertama *pattern* adalah *M*. Pada nilai indeks ke 3, terjadi kecocokan yakni karakter teks *M* dan *pattern M*.
   b. Karena terjadi ketidakcocokan, selanjutnya dilakukan pergeseran *pattern P* dengan jumlah pergeseran sesuai dengan nilai pinggiran *pattern P* yang cocok. Pada kasus ini, karakter pada *pattern* dan teks yang menemukan kesamaan ada di indeks 2-6, maka panjang kesesuaian teks adalah 5 (l= 5).
   c. Nilai pinggiran terpanjang dari *pattern P* yang menemukan kecocokan adalah *P[2…6*], dimana dalam fungsi pinggiran sebelumnya, nilai *pattern* pada indeks urutan 3 dalam fungsi pinggiran adalah 3 (*b*(3)=3).
   d. Setelah dihitung dan dtentukan nilai fungsi pinggiran dan panjang dari kecocokan teks dan *pattern,* maka pergeseran karakter dilakukan dengan memperhatikan rumus berikut:

NILAI PERGESERAN = *l* – *b* (1)

Dimana :

Nilai pergeseran adalah nilai yang akan digeser untuk tiap karakternya

l adalah *p*anjang kecocokan karakter antara *pattern* dan teks

b adalah nilai dalam fungsi pinggiran

Berdasar pada rumus perhitungan diatas, maka dapat ditentukan bahwa nilai pergeserannya adalah 5-3 =2 karakter, sehingga terjadi pergeseran karakter sebanyak 2 karakter ke sebelah kanan. Hasil dari pergeseran ini diperlihatkan dalam Tabel 5 di bawah ini.

Tabel 5. Tabel hasil pergeseran

Indeks T	0	1	2	3	4	5	6
Teks T	S	T	M	I	K	K	A
Indeks *Pattern* P	Nilai Pergeseran		0	1	2	3	4
Pattern P			M	I	K	K	A

4. Hasil dan Pembahasan

4.1 Evaluasi Desain

Dalam alur penelitian yang dilakukan oleh peneliti terdapat tahapan evaluasi desain. Proses evaluasi desain dilakukan untuk mengetahui apakah desain antar muka telah sesuai dengan prinsip kerja algoritma KMP atau belum. Apabila desain belum sesuai, maka dilakukan re-desain terhadap antar muka tersebut. Selain itu evaluasi desain dilakukan untuk mengetahui apakah desain memiliki tampilan yang bagus, mudah dioperasikan, mudah dipelajari dan membuat pemakai merasa nyaman. Desain dimulai dari perancangan antar muka menu utama (diperlihatkan dalam Gambar 7), antar muka untuk melakukan proses pencarian dan pencocokan string (diperlihatkan dalam Gambar 8) serta antar muka untuk melakukan format cetak miring (diperlihatkan dalam Gambar 9).

4.2 Uji Coba Aplikasi

Setelah evaluasi desain selesai dilakukan, untuk melihat unjuk kerja dari desain yang telah dirancang maka perlu dilakukan uji coba aplikasi. Proses pengujian aplikasi dilakukan untuk mengetahui hasil dari perancangan aplikasi yang telah dibuat, baik itu dari segi *design interface* (antar muka) maupun sistem pemrogramannya. Proses uji coba terhadap kinerja aplikasi berkaitan dengan ketepatan proses pencocokan *string* menggunakan algoritma KMP, kecepatan dalam pencarian *string*, serta spesifikasi dari jenis sistem operasi yang dapat digunakan untuk menjalankan aplikasi. Dalam proses pengujian aplikasi ini terdapat beberapa hal yang harus diperhatikan salah satunya terkait dengan spesifikasi hardware yang digunakan. Hal ini sangat berpengaruh pada saat aplikasi dijalankan bisa berupa tingkat ketepatan algoritma yang digunakan, maupun kecepatan proses aplikasi pada saat dijalankan. Dalam pengujian aplikasi untuk melakukan proses pencarian dan pemformatan istilah bahasa Inggris ini membutuhkan spesifikasi perangkat keras yang sesuai dengan kebutuhan aplikasi yang dibuat. Pada saat uji coba ini, aplikasi dijalankan oleh peneliti pada kondisi perangkat lunak sebagai berikut:

1) Processor minimal Pentium Core Duo
2) Memory minimal 2 GB
3) Storage tersisa minimal 400 MB

4.3 Evaluasi Keseluruhan

Proses ini dilakukan setelah melakukan tahap uji coba aplikasi. Evaluasi keseluruhan dimaksudkan untuk mengetahui apakah seluruh desain antar muka telah sesuai dan mengetahui keseluruhan jalannya sistem yang diterapkan dalam aplikasi apakah sudah sesuai dengan prinsip kerja dari algoritma KMP serta mengevaluasi tiap kekurangan yang ada dalam hasil implementasi. Berdasarkan pada evaluasi keseuruhan ini hasil yang didapatkan adalah bahwa aplikasi dapat menerapkan dengan tepat prinsip dari algoritma KMP yaitu mencari dan menemukan *string* yang susunannya sama tepat dengan *pattern*. Kekurangan yang didapatkan adalah

apabila ada kesalahan pengetikkan *string* yang secara kasat mata merupakan istilah bahasa Inggris, maka aplikasi tidak dapat mendeteksi, termasuk apabila ada *string* istilah bahasa Inggris yang tidak terdapat dalam *database* kamus kosakata bahasa Inggris.

4.4 Implementasi Perancangan Basis Data

Implementasi perancangan basis data dilakukan untuk menentukan struktur basis data yang sesuai dengan kebutuhan sistem digunakan dalam penelitian ini. Beberapa tabel dibuat untuk memenuhi kebutuhan sistem, diantaranya tabel kosakata yang berguna untuk menyimpan daftar kosakata istilah bahasa Inggris yang akan dijadikan *pattern* yang akan dibandingkan dengan *string* istilah bahasa Inggris dalam naskah. Gambar 6 dibawah ini menunjukkan contoh tabel kosakata bahasa Inggris yang digunakan.

Kata	arti	sts
Ability	Kemampuan	A
Accounting	Akuntansi	A
Addition	Penambahan	A
Adequate	Memadai	A
Administrators	Administrator	A
Advances	Muka	A
Analysis	Analisis	A
Analysts	Analis	A
Analyze	Menganalisis	A
Annual	Tahunan	A
Applicant	Pemohon	A
Applications	Aplikasi	A
Architects	Arsitek	A

Gambar 6 Contoh tabel kosakata

4.5 Implementasi Perancangan Perancangan Antar Muka

Implementasi selanjutnya setelah *database* kosakata bahasa Inggris siap adalah merancang tampilan antar muka aplikasi. Gambar 7 dibawah ini memperlihatkan tampilan antar muka menu utama.

Gambar 7 Tampilan menu utama

Selain tampilan menu utama, inti dari penelitian ini adalah bagaimana aplikasi dapat menerapkan algoritma KMP untuk melakukan pencarian dan pencocokan *string* istilah bahasa Inggris. Gambar 8 memperlihatkan antar muka pencarian dan pencocokan *string* tersebut.

Gambar 8 Tampilan antar muka pencarian dan pencocokan *string*

Dari Gambar 8 dapat dijelaskan bahwa untuk melakukan proses pencarian dan pencocokan *string* istilah bahasa Inggris, pengguna dapat menuliskan teks di dalam kotak *memo* atau mengambil dari *file* dokumen yang bertipe .doc atau .docx. Sebagai contoh, dalam *form* tersebut terdapat teks **"Belajar membuat Entity Relationship Diagram tentang transaksi Order Menu"**. Sebagai *pattern* adalah daftar kosakata bahasa Inggris yang di *load* dari *database*. Aplikasi akan membandingkan apakah terdapat kesesuaian antara kosakata dalam basis data dengan istilah yang terdapat dalam teks. Gambar 9 memperlihatkan hasil apabila terjadi kecocokan.

Gambar 9 Hasil pencocokan *string* istilah bahasa Inggris

Tampilan Gambar 9 memperlihatkan kondisi apabila suatu *string* cocok dibandingkan dengan *string* kosakata yang terdapat dalam basis data. Proses diawali dengan menekan tombol **Cari Kata & Proses.** Warna *background* teks kuning menunjukkan bahwa *string* tersebut cocok dengan yang terdapat dalam basis

data. Gambar 10 berikut ini memperlihatkan potongan baris program untuk melakukan pencarian dan pencocokan *string*.

```
Tfmatching | Tfmatching.cxButton2Click
procedure Tfmatching.cxButton3Click(Sender: TObject);
begin
  v.First;
  while not v.Eof do
  begin
    Edit2.Text := vKata.AsString;
    SearchForText_AndSelect(RichEdit1, Edit2.Text);
    RE_SetSelBgColor(RichEdit1, clYellow);
    Application.ProcessMessages;
    Sleep(1500);
    v.Next;
  end;
  RichEdit1.SelStart := 0;
  Edit2.Clear;
    {$REGION 'pencarian dan formating'}
    if MessageDlg(
        'Proses pembandingan kata selesai. Apakah langsung di format italic ?',
        mtConfirmation, [mbYes, mbNo], 0) = mrYes then
      begin
        ForeignTermSearch(RichEdit1, Memo1);
        MarkThisWord(RichEdit1, Memo1.Text, clRed, [fsBold]);
      end;
  {$ENDREGION}
end;
```

Gambar 10 Potongan kode pencarian dan pencocokan

Dalam potongan kode yang diperlihatkan dalam Gamabr 10, terdapat beberapa *function* yakni ***SearchForText_AndSelect*** yang digunakan untuk mencari dan membandingkan kata serta ***MarkThisWord*** yang digunakan untuk menandai kata yang cocok. Gambar 11 dan 12 memperlihatkan baris kode tiap-tiap *function* tersebut.

```
function SearchForText_AndSelect(RichEdit: TRichEdit; SearchText: string)
  : Boolean;
var
  StartPos, Position, Endpos: integer;
begin
  StartPos := 0;
  with RichEdit do
  begin
    Endpos := Length(RichEdit.Text);
    Lines.BeginUpdate;
    while FindText(SearchText, StartPos, Endpos, [stMatchCase]) <> -1 do
    begin
      Endpos := Length(RichEdit.Text) - StartPos;
      Position := FindText(SearchText, StartPos, Endpos, [stMatchCase]);
      Inc(StartPos, Length(SearchText));
      SetFocus;
      SelStart := Position;
      SelLength := Length(SearchText);
    end;
    Lines.EndUpdate;
  end;
end;
```

Gambar 11 Kode *function SearchForText_AndSelect*

```
procedure MarkThisWord(RE: TRichEdit; TheWord: String; Color: TColor;
  Style: TFontStyles);
var
  i, CharPos, noChars: integer;
begin
  CharPos := 0;
  noChars := 0;
  for i := 0 to Pred(RE.Lines.Count) do
    noChars := noChars + Length(RE.Lines[i]);
  CharPos := RE.FindText(TheWord, CharPos, noChars, [stWholeWord]);
  RE.SelStart := CharPos;
  RE.SelLength := Length(TheWord);
  RE.SelAttributes.Color := Color;
  RE.SelAttributes.Style := Style;
end;
```

Gambar 12 Kode *function MarkThisWord*

Setelah kata ditemukan dan ditandai, maka kemudian dilakukan pemformatan *italic* (cetak miring) terhadap kumpulan kata yang ditemukan. Gambar 13 memperlihatkan tampilan hasil cetak miring tersebut.

Gambar 13 Hasil pemformatan *italic*

```
procedure ForeignTermSearch(RE: TRichEdit; MM: TMemo);
 var
   RemainingWords, CurrentWord: WideString;
   positionofspace: integer;  i: integer;
 begin
   RemainingWords := RE.Text + ' ';
   RE.Clear;
   while Length(RemainingWords) > 0 do
   begin
     positionofspace := Pos(' ', RemainingWords);
     CurrentWord := Copy(RemainingWords, 1, positionofspace - 1);
     RemainingWords := Copy(RemainingWords, positionofspace + 1, Length
         (RemainingWords));
     RE.SelStart := RE.GetTextLen;
     for i := 0 to MM.Lines.Count do
       if CurrentWord = MM.Lines[i] then
       begin
         RE.SelAttributes.Style := [fsItalic];
         break;
       end
       else
         RE.SelAttributes.Style := [];
     RE.SelText := CurrentWord + ' ';
   end;
 end;
```

Gambar 14 Baris kode untuk cetak miring

Gambar 13 adalah kelanjutan dari Gambar 9. Pada saat pengguna menekan tombol YES yang terdapat dalam kotak konfirmasi cetak miring, maka secara otomatis *string* istilah bahasa Inggris yang telah ditandai tersebut akan tercetak miring. Hasilnya pun dapat disimpan ke dalam basis data dan dapat dibuka kembali. Gambar 14 memperlihatkan potongan baris *function* untuk melakukan cetak miring (*italic*).

Catatan : untuk demo project silahkan dilihat pada alamat berikut: https://youtu.be/eeY4GCCeoWk

5. Kesimpulan

Berdasarkan pada hasil analisis dan implementasi hasil penelitian maka dapat diambil beberapa simpulan dan saran sebagai berikut.

5.1 Simpulan

Beberapa simpulan yang dapat ditarik dari hasil penelitian ini antara lain:

1) Berdasarkan pada hasil analisis yang telah dilakukan, algoritma Knuth Morris Pratt yang merupakan jenis algoritma *exact string matching* dapat diterapkan untuk mencari dan mencocokan *string* secara tepat yang memiliki kesamaan dengan *pattern* yang ditentukan. Hasil dari aplikasi yang dibangun dalam penelitian ini menunjukkan bahwa algoritma KMP dapat menemukan dan mencocokan dengan tepat antara *string* istilah bahasa Inggris yang ada dalam dokumen dengan *string pattern* yang berada dalam *database* kamus kosakata dimana *string* yang dicocokkan memiliki susunan yang sama dengan *pattern* yang ada.
2) Implementasi dari aplikasi yang dihasilkan dalam penelitian menunjukkan bahwa untuk proses pencarian kata yang memiliki susuan yang sama antara teks dengan *pattern*, algoritma KMP yang digunakan ini memiliki tingkat ketepatan yang tinggi. Namun, apabila teks yang dibandingkan dengan *pattern* memiliki susunan yang tidak sama, maka tingkat keberhasilan sangat kecil karena tidak dapat melakukan pendeteksian lebih mendetail. Hal ini dikarenakan sifat dari algoritma KMP yang bersifat *exact string matching*. Selain itu, apabila terdapat *string* istilah bahasa Inggris yang tidak ada dalam kamus akan tetapi terdapat di dalam naskah, maka proses tidak dapat berjalan normal, karena sistem hanya membandingkan *pattern* yang terdapat dalam *database* saja.

5.2 Saran

Beberapa saran yang bisa penulis berikan dalam penelitian ini antara lain :

1) Pada saat proses pencarian berlangsung, apabila ada sebagian kata dari serangkaian kata yang terdeteksi cocok, dalam aplikasi ini masih belum bisa tertandai sehingga untuk peneliti lain yang membahas tema yang sama dapat menambahkan fasilitas ini.
2) Apabila ditemukan kata yang salah dalam penulisan tapi memiliki makna yang sama, dalam penelitian ini belum membahas cara perbaikannya. Untuk itu disarankan bagi peneliti selanjutnya dapat menambahkan pembahasan masalah ini.
3) Aplikasi yang dihasilkan dalam penelitian inimasih belum terdapat fitur untuk mengatasi apabila ada *string* istilah bahasa Inggris yang ada di dalam dokumen tetapi tidak ada dalam *database*. Untuk itu diharapkan pada penelitian selanjutnya dapat menambahkan kekurangan dari hasil penelitian ini.

6. Daftar Rujukan

[1] Wulan, S. 2011. Analisis Penerapan String Matching dalam Komparasi Data Kepesertaan Jaminan Kesehatan Masyarakat (JAMKESMAS). http://repository.uinjkt.ac.id/dspace/bitstream/123456789/2406/1/SRI%20 WULAN-FST.pdf. Diakses pada tanggal 21 November 2016.

[2] http://www.cs.cmu.edu/afs/andrew.cmu.edu/course/15/354/www/postscript/kmp.pdf. diakses pada tanggal 15 desember 2016.

[3] Lestari, S.; Djaya, A., 2011. Aplikasi Search Engine Menggunakan Algoritma Knuth-Morris-Pratt (KMP). MMT-ITS,*Seminar Nasional Manajemen Teknologi XIII*,Surabaya, 5 Februari 2011.
[4] Wibowo, T.; Wibowo, A.; Sari, R.P., 2012. Pembuatan Aplikasi Untuk Mendeteksi Kebenaran Perintah Sql Query Menggunakan Metode Knuth-Morris Pratt (KMP). Jurnal Teknik Informatika. Vol 1 September 2012.
[5] Zaid, I.M.A.; Rayyes, E.K.E., 2012. Parallel Search Using KMP Algorithm in Arabic String. International Journal of Science and Technology. Volume 2 No.7, July 2012. ISSN 2224-3577.
[6] Sa'diah, H., 2017. Implementasi Algoritma *Knuth-Morris-Pratt* Pada Fungsi Pencarian Judul Tugas Akhir *Repository*. JURNAL KOMPUTASI. Vol.14, No.1, January 2017, pp. 115-124. ISSN: 1693-7554.
[7] Waruwu, F.,T.; Mandala, R. 2016. Perbandingan Algoritma Knuth Morris Pratt Dan Boyer Moore Dalam Pencocokan String Pada Aplikasi Kamus Bahasa Nias. Jurnal Ilmiah INFOTEK, Vol 1, No 1, Februari 2016. ISSN 2502-6968.
[8] Rossaria, M.; Susilo, B.; Ernawati., 2015. Implementasi Algoritma Pencocokan *String Knuthmorris-Pratt* Dalam Aplikasi Pencarian Dokumen *Digital* Berbasis Android. Jurnal Rekursif, Vol. 3 No.2 November 2015. ISSN 2303-0755.
[9] Kadir, A. 2012. *Algoritma dan Pemrograman Menggunakan Java.* Yogyakarta: Andi.
[10] Cormen, T.H.; Leirserson, C.; Rivest, R.L., 1990. Introduction to Algorithms. McGraw-Hill Book Company.
[11] Munir, R., 2007. Diktat Kuliah IF2251 *Strategi Algoritmik*. Institut Teknologi Bandung.
[12] Sarno, R.; Anistyasari, Y.; Rahimi, 2012. Semantic Search. Yogyakarta : Andi.
[13] Basee, S., 2000. Computer Algorithms : Introduction To Design And Analysis.
[14] Syaroni, M.; Munir, R., 2005. Pencocokan String berdasarkan Kemiripan Ucapan (Phonetic String Matching) dalam Bahasa Inggris. Seminar Nasional Teknologi Informasi. Yogyakarta. 18 Juni 2005.
[15] Effendi, D.; Hartono, T.; Kurnaedi, A., 2012. Pengembangan Algoritma Boyer Moore pada Translator Bahasa Pemrograman. Jurnal Informatika Universitas Kristen Maranatha. Bandung.
[16] Munir, R., 2004. Pengolahan Citra Digital dengan Pendekatan Algoritmik. Bandung : Informatika.
[17] Sjukani, M., 2010. Algoritma (Algoritma & Struktur Data 1) dengan C, C++, dan Java. Jakarta : Mitrawacanamedia.
[18] Sunni, I., 2010. Music Finder Menggunakan Algoritma KMP Extension. http://www.in*Form*atika.org/~rinaldi/Stmik/20102011/Makalah2010/MakalahStima2010-096.pdf. Diakses 01 Desember 2012.
[19] Ambika. et al, 2013. An Enhanced Version of Pattern Matching Algorithm using Bitwise XOR Operation. International Journal of Computer Applications.
[20] Munir, R., 2007.Diktat Kuliah IF2251 Strategi Algoritmik. Bandung: Program Studi Teknik Informatika. Sekolah Teknik Elektro dan Informatika. ITB.

Halaman ini sengaja dikosongkan