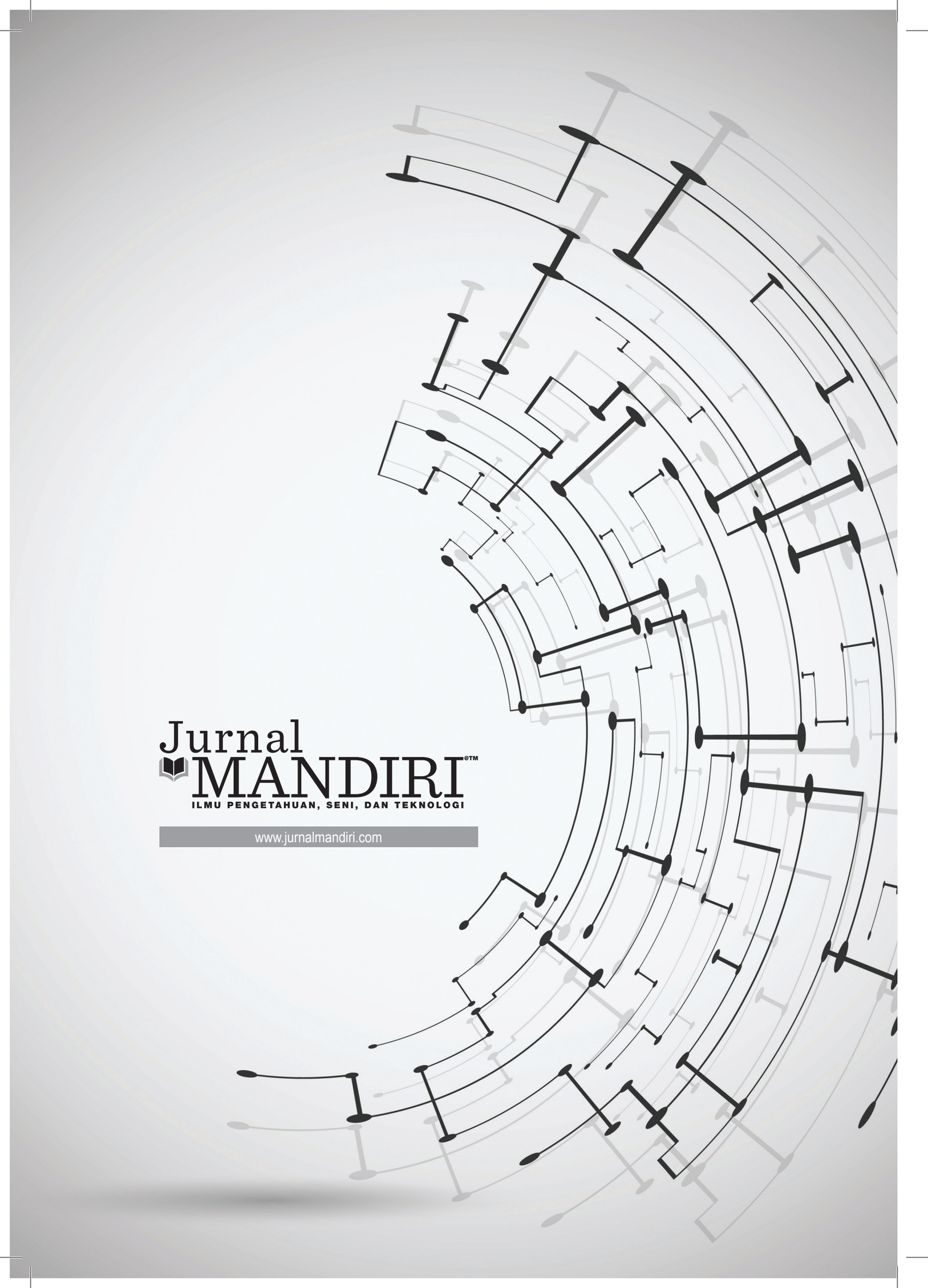


Jurnal MANDIRI©™

ILMU PENGETAHUAN, SENI, DAN TEKNOLOGI

www.jurnalmandiri.com

ISSN : 2580-3220, E-ISSN : 2580-4588
J. Mandiri., Vol. 1, No. 2, Desember 2017 (330 - 352)

Jurnal MANDIRI
ILMU PENGETAHUAN, SENI, DAN TEKNOLOGI

KOMPARASI ALGORITMA *STRING MATCHING* PADA POLA TEKS

Achmad Nur Sholeh
Universitas Pamulang
dosen01531@unpam.ac.id

ABSTRAK

String matching merupakan salah satu algoritma pencocokan kata (*text processing*). Deteksi kemiripan dalam menemukan *pattern* teks suatu informasi berupa "abstrak" karya ilmiah/skripsi mahasiswa menjadi sulit dilakukan jika abstrak tersebut memiliki banyak karakter sehingga prosesnya menjadi lebih rumit dan memerlukan banyak waktu. Implementasi algoritma *string matching* pada prototipe aplikasi deteksi kemiripan pola teks menggunakan bahasa pemrograman Ruby versi 2.1.1 dan *framework Rails* versi 4.1.1. Metode penelitian yang digunakan yaitu metode rekayasa menggunakan model pengembangan sistem *Rapid Application Development* yang terdiri dari tiga fase yaitu, *requirement planning, workshop design,* dan *implementation.* Proses analisa kebutuhan menggunakan metode UML (*Unified Modeling Language*). Penelitian ini bertujuan untuk mencari algoritma *string matching* yang efektif dengan mengkomparasi dua algoritma *string matching* yaitu *Rabin Karp* dan *Aho-Corasick*. Hasil penelitian berupa nilai persentase kemiripan pola teks pada nilai akurasi (*percentage similarity*) dan kecepatan waktu proses (*time complexity*) dari masing-masing algoritma.

Kata kunci : *String Matching, Text Processing, Pattern, Kemiripan Pola Teks, Rabin Karp, Aho-Corasick.*

PENDAHULUAN

Dalam dunia pendidikan perguruan tinggi membuat tugas akhir atau karya ilmiah merupakan salah satu syarat untuk mendapatkan gelar kesarjanaan. Seiring berkembangnya teknologi internet memberi kemudahan penggunanya mendapatkan sumber-sumber informasi karya ilmiah/skripsi, kemudahan akses tidak seimbang dengan "*mental negative*" keterbukaan informasi sehingga membuka jalan atau peluang untuk melakukan berbagai macam bentuk tindak pelanggaran seperti penjiplakan sebuah karya ilmiah "*unoriginaly published*".

Plagiat diduga dilakukan oleh seorang dosen merupakan fenomena yang terjadi di perguruan tinggi negeri pada bulan Juni 2012 (Kompas, 2012). "plagiat tersebut dilakukan dosen dengan cara menjiplak skripsi mahasiswa", dikutip dari halaman kompas.

Algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang tersusun secara sistematis (Munir, 2001). Algoritma *string matching* merupakan algoritma pencocokkan *string,* diantaranya algoritma *Knuth-Morris-Pratt, Brute-Force, Boyer-Moore, Rabin Karp*, *Aho-corasick*, *Manber,* dan algoritma lainnya. Penerapan

algoritma *string matching* diharapkan menjadi solusi nyata dalam mendeteksi pola teks tertentu.

Peneliti menganalisa dua algoritma *string matching* yaitu *rabin karp* dan *aho-corasick* untuk mendapatkan algoritma *string matching* yang efektif dan efisien terhadap kompleksitas waktu (*time complexity)* dan akurasi (*percentage similarity).*

Berdasarkan latar belakang yang telah dikemukakan, maka peneliti tertarik untuk mengambil tema "**Komparasi Algoritma *String Matching* Pada Pola Teks**"

RUMUSAN MASALAH

1. Bagaimana menerapkan Algoritma *String Matching* pada prototipe aplikasi kemiripan pola teks yang dapat disesuaikan dengan kebutuhan pengguna?
2. Bagaimana prosedur kerja algoritma *string matching Rabin Karp* dan *Aho-Corasick* dalam mendeteksi pola teks?
3. Mengetahui algoritma mana yang lebih efektif dan efisien?
4. Bagaimana mengimplementasikan bahasa pemrograman *Ruby 2.1.1* untuk membuat prototipe aplikasi kemiripan pola teks?

PEMBATASAN MASALAH

1. Peneliti menggunakan dua buah Algoritma *String Matching*, yaitu algoritma *Rabin Karp,* dan algoritma *Aho-Corasick*.
2. Bahasa pemrograman yang digunakan *Ruby 2.1.1* dan *framework rails* versi 4.1.1.
3. Sampel data penelitian yang digunakan adalah teks pada halaman abstraks karya ilmiah mahasiswa.
4. Prototipe aplikasi ini tidak menggunakan sistem database.
5. Penentuan data teks abstraks yang diuji dilakukan secara manual.

TUJUAN PENELITIAN

1. Membangun prototipe aplikasi kemiripan pola teks dengan menerapkan Algoritma *String Matching*.
2. Mengkomparasi kinerja algoritma *Rabin Karp* dan algoritma *Aho-Corasick*.

TINJAUAN TEORITIK

Algoritma

Menurut Suarga teknik penyusunan langkah-langkah penyelesaian masalah dalam bentuk kalimat dengan jumlah kata terbatas, tetapi tersusun secara logis dan sistematis. Suatu prosedur yang jelas untuk menyelesaikan suatu persoalan dengan menggunakan langkah-langkah tertentu dan terbatas jumlahnya (Suarga, 2004).

Terdapat beragam klasifikasi algoritma dan setiap klasifikasi mempunyai alasan tersendiri. Salah satu cara untuk melakukan klasifikasi jenis-jenis algoritma adalah dengan memperhatikan paradigma dan metode yang digunakan untuk mendesain algoritma tersebut. Beberapa paradigma yang digunakan dalam menyusun suatu algoritma, sebagai berikut :

1. *Divide and Conquer*, Suatu teknik yang melakukan dekomposisi dari masalah utama, dan menyelesaikan masalah tersebut dengan cara menyelesaikan sub-sub masalah yang sejenis dengan masalah utama, dan menggabungkan solusi-solusi dari sub masalah untuk mendapatkan solusi masalah utama.
2. *Dynamic Programming,* Teknik *dynamic programming* memiliki prinsip yang sama dengan *devide and conqueror*, dimana keduanya menyelesaikan suatu masalah dengan cara memecahnya menjadi *sub-sub* masalah yang dapat diselesaikan secara rekursif. Meskipun demikian, pada *dynamic programming* diusahakan menghindari dilakukannya redudansi proses suatu *sub* masalah (selama proses rekursif), yakni dengan cara mencatat hasil yang telah diperoleh dari suatu sub masalah pada suatu tabel, dengan demikian untuk proses *sub* masalah lain yang lebih besar, tidak perlu menghitung berulang-ulang nilai

sub masalah yang lebih kecil yang pernah dihitung sebelumnya.

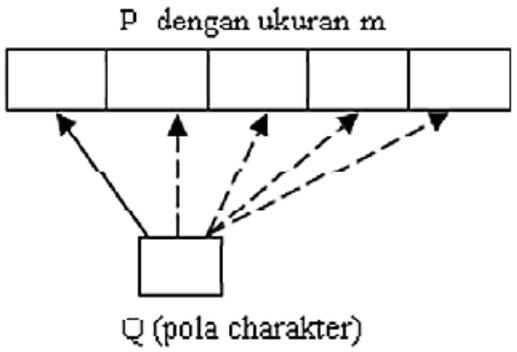
3. Greedy, Sebuah paradigma greedy mirip dengan sebuah algoritma dynamic programming, bedanya jawaban dari sub masalah tidak perlu diketahui dalam setiap tahap dan menggunakan pilihan "serakah" apa yang dilihat terbaik pada saat itu. Teknik algoritma ini biasanya digunakan untuk masalah-masalah optimasi, seperti *job scheduling, shortest path* pada *graph*.

Dalam hal ini algoritma *string matching* atau pencocokkan *string* termasuk ke dalam jenis algoritma *dynamic programming*.

String Matching

Menurut Black (Syaroni, 2004) *string* adalah susunan dari karakter-karakter (angka, alfabet atau karakter yang lain) dan biasanya direpresentasikan sebagai struktur data Array. *String* dapat berupa kata, frase, atau kalimat. Black juga memberikan definisi mengenai *String Matching* sebagai sebuah permasalahan untuk menemukan pola susunan karakter *string* di dalam *string* lain atau bagian dari isi teks.

Menurut Stephen (Arliadinda, 2005) Algorima *string matching* mengikuti cara kerja pencarian buku. Algoritma ini membaca teks dengan bantuan sebuah *temporary* yang mempunyai ukuran (lebar) adalah m. Untuk memudahkan, *temporary* ini diinisialisasikan dengan huruf P. Setelah itu dibutuhkan sebuah *temporary* lagi, diinisialisasikan dengan huruf Q dengan panjang n, yang berisi pola karakter yang akan dicari. Kemudian algoritma ini akan membandingkan P dengan Q. Cara kerja ini dinamakan *attempt* dan apabila pencocokan tersebut tidak sesuai, maka P akan mengeser ke sebelah kanan, kemudian akan dicocokan kembali dengan Q. Demikian seterusnya hingga pada P terakhir yang ada di sebelah kanan (akhir dari teks). Mekanisme ini biasa disebut dengan *sliding window mechanism*.



Gambar 1. Pencocokan *String*

Berdasarkan arah pemeriksaan karakter, metode yang digunakan dikelompokkan menjadi :

1. Metode pembacaan berawal dari posisi kiri mengarah ke kanan (*from left to right*)
2. Metode pembacaan berawal dari posisi kanan mengarah ke kiri (*from right to left*).
3. Metode pencarian dengan aturan tertentu (*in specific order*).
4. Metode yang tidak memiliki suatu pola tertentu (*in any order).*

Seluruh algoritma *string matching* merupakan algoritma yang memecahkan persoalan pencarian *string*, dimana algoritma tersebut menghasilkan pola *string* yang terdapat pada *string* teks dengan data masukan minimal sebagai berikut :

1. Sebuah teks, yaitu sebuah rangkaian *string* dengan panjang n karakter dengan pola persamaan y = y[0..n-1]
2. *Pattern*, yaitu sebuah *string* dengan panjang m karakter dimana panjang *pattern* tidak lebih panjang dari panjang teks yang akan dicari dengan pola persamaan x = x[0..m-1].

Kedua *string* tersebut dibentuk dari set karakter yang disebut alphabet dinotasikan oleh Σ (sigma) dengan ukuran σ (teta) (Atmoprawiro, 2006).

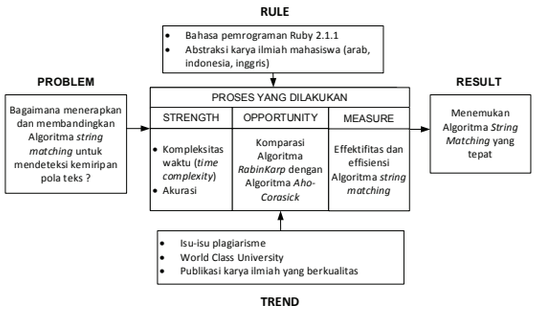
Rabin Karp

Algoritma Rabin Karp dibuat oleh Michael O. Rabin dan Richard M. Karp. Algoritma ini lebih berguna pada pencarian *multiple pattern*

dari pada pencarian *single pattern*. Karena algoritma ini tidak memperdulikan huruf besar atau kecil, dan tanda baca yang digunakan. Algoritma Rabin Karp ini menggunakan fungsi *hash*. Fungsi *hash* adalah fungsi yang digunakan untuk mengubah *string* menjadi *untaian integer*. Pada algoritma ini untaian *string* akan diubah menjadi *integer* berdasarkan bilangan ASCII-nya. Karena menggunakan bilangan ASCII, proses komputasi menjadi lebih "dekat" kebahasa mesin. Pendekatan utamanya adalah, *string* yang sama akan memiliki nilai *hash* yang sama (Wirawan, 2008).

Berikut ini akan dijelaskan bagaimana algoritma Rabin Karp melakukan pencarian kata (Baedlowi, 2006).

1. Asumsikan teks adalah *string* t yang panjangnya m dan *pattern* (*string* yang akan dicari) P panjangnya n.
2. Asumsikan Si menyatakan sebuah *substring* dengan panjang n, yang berkelanjutan pada awal teks misalkan S0 adalah *substring* dengan panjang n diawal t.
3. Ide utama dari algoritma ini adalah memanfaatkan fungsi *hash* untuk memetakan setiap Si kedalam himpunan. Fungsi *hash* digunakan untuk menempatkan suatu *record* yang mempunyai nilai kunci k. Fungsi *hash* yang paling umum berbentuk H(k)=kmodm. Didalam algoritma ini k dapat berupa *string* P atau *string* Si.
4. Sketsa Algoritma :
   a. Pertama kita menghitung nilai fungsi *hash* dari *string* P.
   b. Kemudian untuk masing-masing Si kita menghitung fungsi *hash*- nya.
   c. Lakukan penelusuran terhadap Si, jika h(Si) = h(P), maka lakukan pencocokan antara *string* Si dengan *string* P secara brute force.
   d. jika h(Si) ≠ h(P) kita tidak perlu melakukan pencocokan *string*, penulusuran dilanjutkan kembali terhadap Si yang berikutnya sampai ditemukan atau sampai *string* t berakhir. *Pseudocode* dari algoritma ini dituliskan sebagai berikut :



Gambar 2. *Pseudocode Rabin Karp*

Teori ini jarang digunakan untuk mencari kata tunggal, namun teori ini cukup penting dan sangat efektif bila digunakan untuk mencari lebih dari satu kata. Kasus terbaik dari algoritma ini bisa mencapai O(n), dimana n adalah panjang teks. Sayangnya kasus terburuk dari algoritma ini adalah O(nm), dimana m adalah panjang kata yang menjadi sebab mengapa jarang digunakan. Keuntungan tersendiri dari algoritma ini adalah dapat mencari k kata dalam waktu rata-rata O(n), yang tidak bergantung pada ukuran k.

Aho-Corasick

Algoritma *Aho-Corasick* adalah algoritma pencarian *string* atau sering juga pencocokan *string* (*string matching algorithm*) yang ditemukan oleh Alfred V. Aho dan Margaret J. Corasick (Corasick, 1975). Algoritma ini merupakan algoritma penyesuaian kamus yang menempatkan elemen dalam kumpulan *string* yang terhingga. Algoritma ini menyesuaikan semua pola secara bersamaan. Kompleksitas algoritma ini adalah O(n+m+z), dengan n merupakan banyak pola, m merupakan panjang dari teks yang digunakan dalam pencarian, dan z merupakan jumlah output yang sesuai atau jumlah kemunculan pola. Algoritma *Aho-Corasick* pertama-tama akan membuat mesin *automata* yang menyerupai *trie* dengan *link* tambahan diantara node internal dari keyword atau pola yang ada. Link tambahan ini memungkinkan

transisi yang cepat saat terjadi kegagalan dalam proses pencocokan pola sehingga *automata* dapat berpindah ke cabang *trie* yang lain yang memiliki *prefix* yang mirip. Dengan adanya link tambahan tersebut, *automata* dapat berpindah saat proses pencocokan pola tanpa diperlukannya *backtracking* (Handoko, 2014).

Trie adalah struktur data *ordered tree* yang dipergunakan untuk menyimpan set yang dinamis atau *array assosiatif* dimana kunci yang ada biasanya berupa *string*. Sebuah *trie* memiliki berbagai kelebihan dibandingkan dengan *binary tree* dan dapat juga dipergunakan untuk menggantikan tabel *hash*.

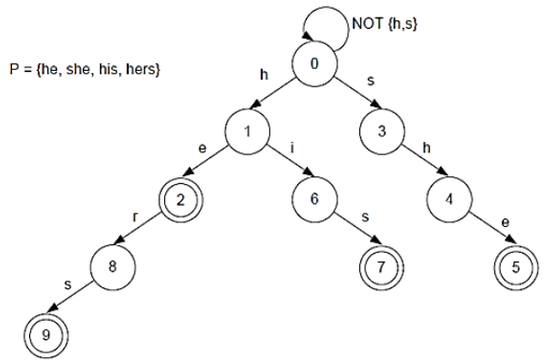
Algoritma *Aho-Corasick* didasarkan pada keyword *trie*. Keyword *trie* untuk himpunan pola P adalah *trie* dengan akar K dimana:

1. Setiap edge K dinamakan dengan sebuah karakter.
2. Dua edge yang keluar dari sebuah simpul memiliki nama yang berbeda.
3. Untuk setiap XÎP terdapat sebuah simpul v dengan L(v)=X
4. Label L(v) dari setiap daun v adalah sama dengan XÎP

Konstruksi keyword *trie* untuk himpunan pola P = {P1,…,Pk} dimulai dari simpul akar atau awal, dengan memasukkan setiap pola Pi satu per satu dengan aturan:

1. Mulailah dari simpul akar dengan mengikuti path yang dinamakan dengan karakter dari Pi.
2. Jika path berakhir sebelum Pi, maka lanjutkan dengan menambah edge baru dan simpul untuk setiap karakter Pi.
3. Simpan pengenal (*identifier*) i dari Pi pada simpul terminal dari path. Pi adalah elemen ke-i dari himpunan P. Proses konstruksi keyword *trie* memerlukan waktu $O(|P1|+...+|Pk|) = O(n)$.

Sebagai contoh, terdapat himpunan pola P = {he, she, his, hers}. Keyword *trie* yang terbentuk dari himpunan pola P dengan *state* terminal dilambangkan dengan lingkaran ganda (Vilo, 2008).



Gambar 3. Contoh Keyword *Trie*

Pencarian sebuah untaian *string* X dimulai dari simpul akar (root) atau awal mengikuti *path* yang diberikan label karakter dari X. Jika penelusuran path berhenti pada simpul dengan *identifier* atau dengan kata lain berhenti pada simpul terminal, maka X adalah keyword dalam kamus atau *trie*. Sebaliknya, jika penelusuran *path* berhenti sebelum akhir dari *string* X, maka *string* tersebut tidak ditemukan dalam kamus (Handoko, 2014).

Selanjutnya keyword *trie* diubah menjadi *automata* untuk mendukung pencocokan *linear-time* dengan setiap simpul dalam keyword *trie* menjadi *state* dalam *automata* dan simpul akar menjadi *state* awal atau *state* 0. Lalu, untuk menentukan perpindahan *state* dalam *automata*, ditambahkan tiga fungsi:

1. Fungsi goto, g(q, a) menghasilkan *state* yang dituju dari *state* saat itu (q) setelah menerima input karakter a.
   a. Jika edge (q, v) diberikan label a, maka g(q, a) = v.
   c. g(0, a) = 0 untuk setiap a yang tidak diberikan sebagai label untuk edge yang keluar dari *state* awal. *Automata* tetap berada pada *state* awal jika menerima input berupa karakter yang tidak dikenal.
   c. Lainnya g(q, a) = Ø
2. Fungsi *failure* f(q) untuk q ≠ 0 untuk *state* saat tidak ditemukan kecocokan. f(q) adalah simpul yang diberikan label *longest proper suffix* w dari L(v) dengan w merupakan *prefix* dari pola tertentu.

3. Fungsi output out(q) memberikan *string* yang dikenal saat memasuki sebuah *state*.

Perubahan keyword *trie* menjadi *automata* dan penambahan fungsi akan menambahkan transisi pada keyword *trie* dari satu simpul ke simpul lain (Vilo, 2008)

BAHASA PEMROGRAMAN RUBY

Bahasa *scripting* berorientasi objek yang diciptakan pertamakali oleh Yukihiro Matsumoto pada awal tahun 1990-an (Lenz, 2008). Ruby adalah bahasa pemrograman berorientasi objek. Tujuan dari ruby yaitu menggabungkan kelebihan dari semua bahasa-bahasa pemrograman *scripting* yang ada di dunia, segala sesuatu yang dimanupulasi di ruby adalah objek yang manipulasi benda itu sendiri adalah objek itu sendiri. (Ruby et al, 2010).

Rails merupakan *framework* pengembangan aplikasi berbasis web ditulis menggunakan Bahasa pemrograman Ruby. *Framework* merupakan kumpulan software, pustaka (*library*), dan perkakas lain (*tools*) yang dirangkai menjadi satu kesatuan dan digunakan untuk membangun aplikasi secara cepat dan mudah dikembangkan. Biasanya dalam *framework* disediakan modul atau *class* yang berfungsi untuk mengakses basis data, *template* tampilan antarmuka pengguna, dan manajemen pengguna. konsep *rails* memiliki keunggulan dibanding yang lain, terutama dalam masalah produktivitas pengembang dan kecepatan implementasi aplikasi. Prinsip utama *rails* dibagi menjadi dua, yaitu : (Rubyon*rails*, 2015)

1. *Don't Repeat Yourself*

Tidak melakukan penulisan kode yang sama berulang kali. Inilah salah satu kelebihan *rails*, yaitu membuat *developer* web sedikit menulis kode program artinya cukup kita buat sekali untuk kemudian digunakan berulang kali. Sebagai contoh dalam Ruby sudah terdapat *Active Record* yang menyediakan definisi fungsi-fungsi basis data dalam bentuk kelas-kelas. Mendefinisikan sendiri *class* hanya untuk melakukan akses ke basis data hanya akan memboroskan waktu *programmer* dan bersifat redundan.

2. *Convention Over Configuration*

Konfigurasi mengikuti konvensi atau aturan yang telah dibuat oleh *rails*, konfigurasi akan ditangani secara otomatis oleh *rails* berdasarkan konvensi tersebut. Namun, *rails* juga memberikan keleluasaan bagi programmer untuk tidak mengikuti konvensi dengan memberikan kode konfigurasi tambahan pada aplikasi.

Rails menggunakan konsep MVC (*Model View Controller*) sehingga memudahkan programmer dalam melakukan manajemen *codes*.

RAPID APPLICATION DEVELOPMENT (RAD)

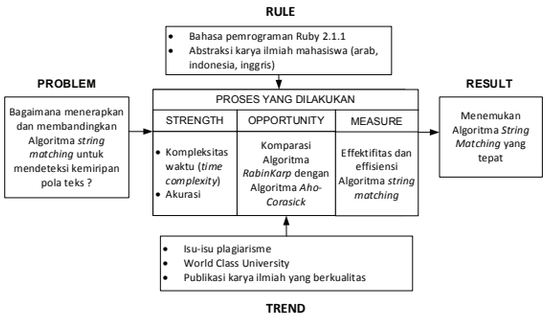
Salah satu metode pengembangan suatu sistem informasi dengan waktu yang relatif singkat. Untuk pengembangan suatu sistem informasi yang normal membutuhkan waktu minimal 180 hari, akan tetapi dengan menggunakan metode RAD suatu sistem dapat diselesaikan hanya dalam waktu 30-90 hari (Agustinus, 2002).

Rapid Application Development (RAD) adalah strategi siklus hidup yang ditujukan untuk menyediakan pengembangan yang jauh lebih cepat dan mendapatkan hasil dengan kualitas yang lebih baik dibandingkan dengan hasil yang dicapai melalui siklus tradisional (Mcleod, 2002).

Rapid Application Development (RAD) merupakan gabungan dari bermacam-macam teknik terstruktur dengan teknik *prototyping* dan teknik pengembangan *joint application* untuk mempercepat pengembangan sistem (Bentley, 2004).

Rapid Application Development (RAD) adalah suatu pendekatan berorientasi objek terhadap pengembangan sistem yang mencakup suatu metode pengembangan serta perangkat-perangkat lunak. RAD bertujuan mempersingkat waktu yang biasanya diperlukan dalam siklus hidup pengembangan sistem tradisional antara perancangan dan penerapan suatu sistem informasi (Kendal, 2010).

KERANGKA PENELITIAN



Gambar 4. Desain kerangka pemikiran

Proiblem yang dihadapi yaitu bagaimana membangun prototipe aplikasi algoritma *string matching* dalam mendeteksi kemiripan pola teks menggunakan bahasa pemrograman *Ruby* versi 2.1.1 dan sampel data yang digunakan yaitu abstraksi karya ilmiah mahasiswa. Dasar pemilihan sampel data merujuk kepada definisi abstrak menurut Moedjiono, abstrak berisi intisari/deskripsi singkat/kondensasi dari naskah laporan penelitian. Isi abstrak bersifat keseimbangan antara penjelasan deskriptif dan informatif (Moedjiono, 2012).

Abstrak merupakan penggambaran secara maya tentang suatu fakta dengan membaca bagian ini pembaca dapat memahami intisari dari suatu karya ilmiah (Kusmana, 2010).

Peneliti menyimpulkan bahwa sampel data berupa abstrak karya ilmiah dapat mewakili keseluruhan dari isi karya ilmiah yang akan digunakan sebagai data pembanding antara karya ilmiah yang satu dengan yang lainnya.

Prototipe aplikasi diharapkan dapat melakukan komparasi Rabin Karp dan Aho-Corasick dengan parameter pengujian terhadap kompleksitas waktu (*time complexity*), dan akurasi (*percentage similarity*) sehingga didapatkan algoritma *string matching* yang tepat.

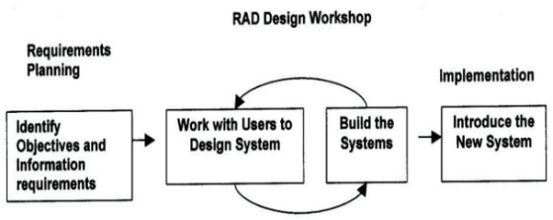
HIPOTESIS PENELITIAN

Berdasarkan masalah pokok serta tujuan yang ingin dicapai dalam penelitian ini, maka dirumuskan hipotesis sebagai berikut: diduga adanya perbedaan kompleksitas waktu dan akurasi dalam menemukan kemiripan pola teks menggunakan algoritma Rabin Karp dan algoritma Aho-Corasick pada sampel data teks abstrak dalam teks bahasa Indonesia, teks bahasa Arab, dan teks bahasa Inggris.

METODE PENELITIAN

Desain Penelitian

Metode RAD digunakan untuk membangun prototipe aplikasi yang dibutuhkan. Alasan digunakannya metode RAD mengacu kepada Martin (Pressman, 2002) *Rapid Application Development* adalah sebuah model proses perkembangan perangkat lunak sekuensial linier yang menekankan siklus perkembangan yang sangat pendek. Beberapa alasan lainnya aplikasi yang dibangun merupakan aplikasi berskala kecil, tidak memerlukan waktu lama, serta tidak memerlukan pemeliharaan. Tujuan lainnya mempersingkat waktu pengerjaan serta proses yang dihasilkan didapatkan secara cepat dan tepat sebuah pendekatan atau model pengembangan perangkat lunak yang mencoba menyederhanakan berbagai tahapan dalam proses pengembangan tersebut sehingga menjadi lebih adaptif dan fleksibel.



Gambar 5. RAD

Menurut Kendall (Kendal, 2008) pendekatan *Rapid Application Development* meliputi fase-fase berikut:

1. **Perencanaan Kebutuhan (*Requirement Planning*).**

Merupakan sebuah tahap awal untuk suatu proyek, dimana pengguna dan analis mengidentifikasi kebutuhan untuk memenuhi tujuan sistem, mengidentifikasi kebutuhan informasi yang timbul dari tujuan tersebut, serta menentukan batasan-batasan sistem yang akan dibuat, kendala serta alternatif masalah.

2. Proses Desain (*Design Workshop*)

Merupakan tahap lanjutan dari tahap perencanaan kebutuhan dimana dilakukan pengidentifikasian dari solusi alternatif yang ada dengan pemilihan solusi terbaik. Setelah itu, dilanjutkan dengan melakukan pemodelan proses bisnis dan desain pemrograman untuk data-data yang telah diperoleh yang nantinya akan dimodelkan dalam arsitektur informasi.

3. Implementasi (*Implementation System*)

Tahap ini terdiri atas dua tahapan, yaitu tahap pengimplementasian sistem ke dalam bahasa pemrograman (*coding*) dan tahap pengujian sistem oleh beberapa *owner*, *analyst*, dan *developer* dengan tujuan apakah sistem yang dibangun sudah berjalan dengan baik pada saat pengoperasiannya atau masih terdapat kesalahan (*error*).

METODE PENGUMPULAN DATA

Studi Pustaka

Bertujuan untuk mencari informasi tentang data-data yang bersifat teoritis yang dijadikan sebagai acuan untuk analisis komparasi algoritma *string matching*. Referensi tersebut berasal dari buku-buku pegangan maupun publikasi hasil penelitian yang berhubungan dengan algoritma *string matching* dalam bentuk jurnal, *paper, prosiding*, makalah, skripsi, ataupun melakukan komunikasi langsung dengan ahli-ahli.

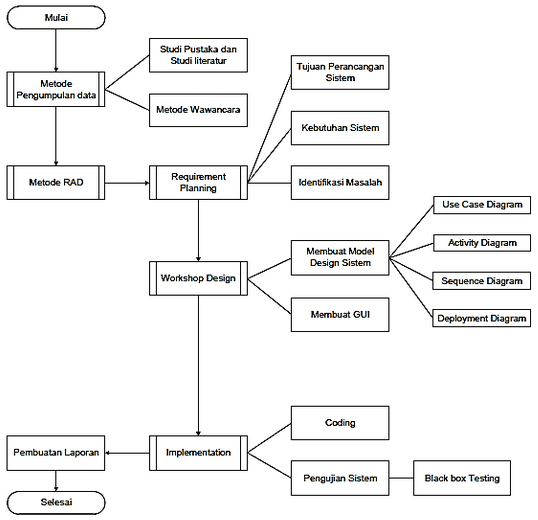
Wawancara

Teknik pengumpulan data yang dilakukan dengan berdiskusi kepada pihak yang dapat memberikan informasi tentang kebutuhan aplikasi yang akan dibuat.

Pemilihan Sampel

Metode pemilihan sampel menggunakan *Sampling Purposive* yaitu sampel yang akan digunakan berupa halaman abstrak pada karya ilmiah mahasiswa dan teknik pengambilan sampelnya adalah *Nonprobability sampling*.

Langkah-Langkah Penelitian



Gambar 6. Langkah penelitian

1. Tahap pengumpulan data dibagi menjadi dua yaitu :
   a. Studi pustaka, mempelajari berbagai referensi. Topik-topik yang dikaji yaitu: Algoritma *string matching*, plagiarisme, teori perbandingan, metode *Rapid Application Development*, bahasa pemrograman Ruby 2.1.1.
   b. Wawancara, pencarian informasi secara langsung kepada narasumber.
2. Metode RAD, merupakan model pengembangan perangkat lunak yang dibagi menjadi tiga tahapan yaitu:
   a. *Requirement Planning*, pada tahap ini system analis bertemu dengan *end-user* untuk mengidentifikasi tujuan perancangan sistem, syarat kebutuhan sistem dan mengidentifikasi masalah yang menjadi latar belakang dalam perancangan sistem.
   b. *Workshop design*, pada tahap ini pembuatan model desain sistem dengan menggunakan *tools* UML meliputi pembuatan *use case, Sequence diagram*, *activity diagram*, dan *Deployment diagram*, serta perancangan desain tampilan aplikasi dalam *Mode Graphical User Interface* (GUI).

c. *Implementation*, pada tahap ini proses analisis dan perancangan sistem dijadikan sebagai acuan untuk penulisan *coding* menggunakan bahasa Ruby 2.1.1 dan kemudian melakukan pengujian menggunakan metode *black box testing*.

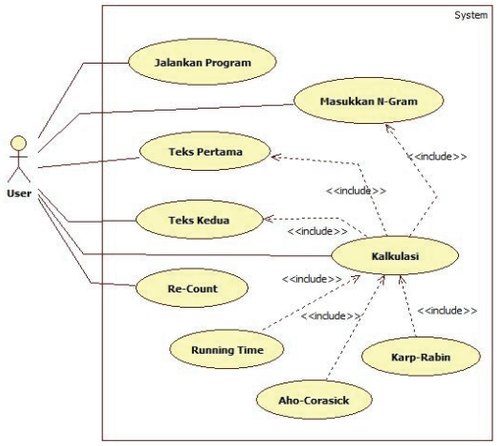
HASIL DAN PEMBAHASAN

Analisis sistem

Proses analisis sistem menggunakan metode RAD dengan model *Unified Modeling Language* (UML).

1. *User case diagram*

   Use case diagram digunakan dalam menggambarkan fungsionalitas yang diharapkan dari sebuah sistem perangkat lunak. yang ditekankan adalah "apa" yang diperbuat oleh sistem bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara *actor* dengan sistem (Munawar, 2005).

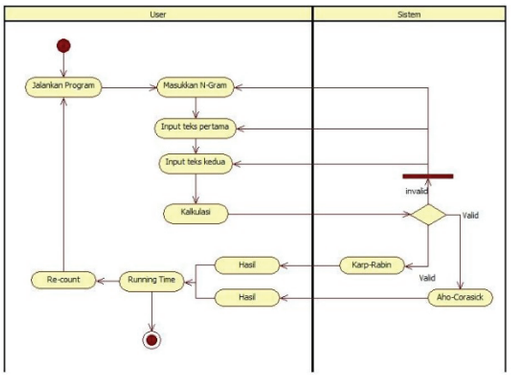


Gambar 7. *Use Case Diagram*

   Use case diagram di atas merupakan *use case* yang mendeskripsikan interaksi *actor User* dengan aplikasi. *Use case diagram* memiliki Sembilan *use case* yaitu: jalankan program, masukkan *N-gram*, teks pertama, teks kedua, kalkulasi, *Rabin Karp, Aho-Corasick, running time,* dan *re-count*.

2. *Activity diagram*

   activity diagram menggambarkan aktifitas-aktifitas yang terjadi di dalam *user* dan sistem. berikut ini adalah *activity* diagram yang terdapat pada sistem.



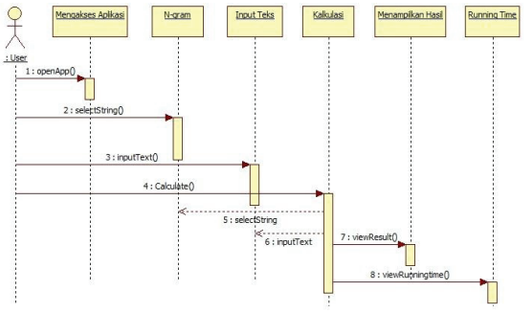
Gambar 8. *Activity Diagram User*

   Aktor yang terlibat dalam aktifitas ini adalah *User*. Aktifitas yang terjadi yaitu *user* menjalankan prototipe aplikasi dengan menentukan *N-gram* kemudian *user* memasukan teks abstrak pada input teks pertama sebagai teks pembanding dan input teks kedua sebagai teks yang dibandingkan. Sistem akan mengecek apakah *n-gram*, kolom teks pertama dan teks kedua sudah diinput dengan benar, jika tidak maka sistem akan memberikan peringatan untuk melengkapi atau mengisinya dengan benar jika semuanya sudah valid kemudian sistem akan memproses menggunakan algoritma Rabin Karp dan Aho-Corasick untuk mendapatkan nilai persentase dan waktu proses yang telah dilakukan oleh kedua algoritma tersebut. *User* dapat mengulang kembali dengan memilih menu *re-count*.

3. *Sequence diagram*

   Sequence diagram digunakan dalam menggambarkan interaksi antar objek didalam dan disekitar sistem berupa message yang digambarkan terhadap waktu. *Sequence diagram* terdiri dari dimensi vertikal (waktu) dan dimensi horizontal (objek-objek terkait). Bisa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah *event* untuk menghasilkan output tertentu (Munawar, 2005). Pada *Sequence diagram* dijelaskan bahwa seorang

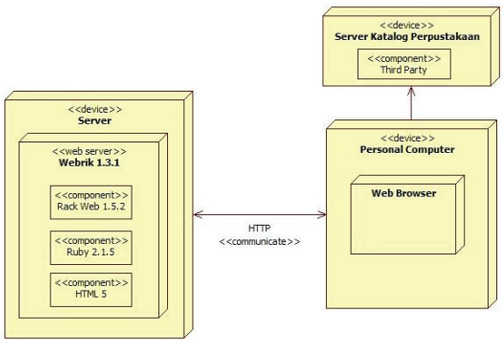
user membuka aplikasi *web browser* dengan alamat http://enigmatic-inlet-8996.herokuapp.com/ untuk dapat mengakses aplikasi yang sudah *dihosting* dan diberi nama *Karp Corasick*. Selanjutnya server akan merespon dengan menampilkan aplikasi tersebut. *User* diminta untuk memasukkan parameter-parameter yang tersedia dalam menu seperti masukkan *N-gram*, Input teks pertama dan input teks kedua. Kemudian sistem akan memvalidasinya untuk dapat melanjutkan proses kalkulasi, jika tidak valid sistem akan menampilkan pesan kesalahan yang harus dilengkapi pengguna.



Gambar 9. *Sequence diagram User*

4. *Deployment diagram*

Deployment diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, dimana komponen akan terletak (pada mesin, server atau perangkat keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisik.



Gambar 10. *Deployment diagram*

PERANCANGAN SISTEM

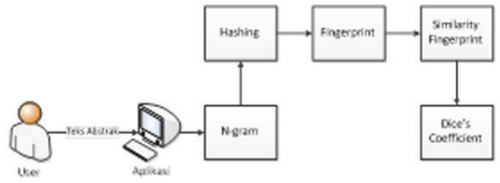
Perancangan sistem ini bertujuan untuk memberikan gambaran proses yang terjadi pada sistem. Proses yang terjadi dibagi menjadi dua yaitu arsitektur aplikasi dan desain aplikasi *Grapichal User Interface* (GUI).

1. Arsitektur Aplikasi

Arsitektur aplikasi merupakan alur proses atau tahapan proses sebuah algoritma dalam mendeteksi *multiple pattern string matching*. Model arsitektur aplikasi yang dibuat dalam penelitian ini sebagai berikut:

A. *Rabin Karp*

Alur proses yang terjadi pada algoritma rabin karp dijelaskan pada gambar berikut :



Gambar 11. Alur Proses Rabin Karp

Teks abstrak yang diinput oleh *user* akan di kelompokkan sesuai nilai *N-gram* yaitu rangkaian *terms* dengan panjang N, kemudian dilakukan proses *hashing* yaitu proses merubah sub*string* menjadi bilangan *integer* atau yang disebut dengan *hash value*, kumpulan dari nilai-nilai *hash* disebut dengan *fingerprint* dari suatu dokumen. kemudian *fingerprint* ini yang akan dijadikan dasar pembanding kesamaan antara teks yang telah dimasukkan dengan persamaan *dice's coefficient*.

B. *Aho-Corasick*

Proses kerja yang terjadi pada algoritma aho-corasick dijelaskan pada gambar berikut:

Gambar 12. Alur Proses Aho-Corasick

User meng-input teks abstrak yang kemudian oleh aplikasi akan dikelompokkan sesuai nilai *N-gram*, kemudian teks pertama dan teks kedua akan dicari pola kemiripannya berdasarkan pengelompokkan tersebut, dan akan dijadikan dasar pembanding kesamaan antara teks yang telah dimasukkan dengan persamaan *dice's coefficient.*

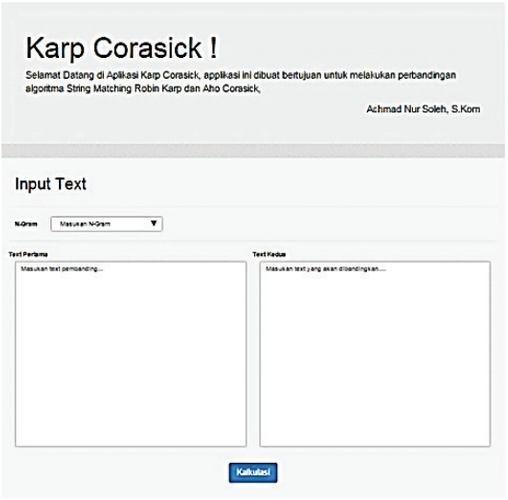
B. *N-gram*

N-gram adalah rangkaian *terms* dengan panjang N. *terms* yang digunakan adalah kata. *N-gram* merupakan sebuah metode yang diaplikasikan untuk pembangkitan kata atau karakter. Metode *N-gram* ini digunakan untuk mengambil potongan-potongan karakter huruf sejumlah n dari sebuah kata yang secara kontinuitas dibaca dari teks sumber hingga akhir dari dokumen. Berikut ini adalah contoh *N-gram* dengan n=4 :

Contoh teks: *N-gram* adalah rangkaian Sehingga dihasilkan rangkaian 4-grams yang diturunkan dari teks: {N-gr}, {-gra}, {gram}, {ram }, {am a}, {m ad}, { ada}, {adal}, {dala}, {alah}, {lah }, {ah r}, {h ra}, { ran}, {rang}, {angk}, {ngka}, {gkai}, {kaia}, {aian}. Berdasarkan penelitian sebelumnya (Pandawa, 2015), pemilihan nilai *n-gram* berdasarkan yang memiliki hasil error kemiripan paling kecil dan waktu proses yang tercepat untuk pemilihan nilai *N-gram*, semakin kecil nilai *N-gram* maka hasilnya akan semakin baik akan tetapi nilai *N-gram*<5 menunjukkan *error* yang besar. Sedangkan untuk *N-gram*=5, nilai errornya tidak terlalu besar dan mendekati nilai error terkecil. Oleh karena itu nilai *N-gram* yang dipilih untuk pengujian selanjutnya adalah *N-gram*=5.

2. Desain GUI

Rancangan tampilan utama prototipe aplikasi kemiripan pola teks. Terdapat menu-menu yang sudah diuraikan sebelumnya pada proses desain model, yaitu nilai *N-gram*, teks pertama sebagai teks yang akan dibandingkan dan teks kedua sebagai teks pembanding dan tombol proses (kalkulasi).



Gambar 13. Desain GUI

IMPLEMENTASI

1. *Rabin Karp*

a. *N-gram*

Berikut ini *source code* yang berfungsi untuk mendapatkan nilai *N-gram* :

```
def initialize options = {}
   @first_text ||= options[:first_text]
   @second_text ||= options[:second_text]
   @ngram ||= options[:ngram] || 2
  End
def first_text_ngram
   @first_text_ngram ||= first_text.downcase.to_ngram(ngram)
  end
def second_text_ngram
   @second_text_ngram ||= second_text.downcase.to_ngram(ngram)
  end
```

Gambar 14. Source code *N-gram*

b. *Hashing*

Berikut ini *source code* yang berfungsi untuk melakukan proses *rolling hash* :

```
private
  def rolling_hash ngram_text
    ngram_text.map {|text| text.base11_hash }
  end
def first_text_hashes
  @first_text_hashes ||= rolling_hash first_text_ngram
end
def second_text_hashes
  @second_text_hashes ||= rolling_hash second_text_ngram
end
```

Gambar 15. *Source code hash*ing

c. *Fingerprint*

Berikut ini *source code* yang berfungsi untuk melakukan proses *Fingerprint* :

```
def first_text_fingerprints
  @first_text_fingerprints ||= first_text_hashes.uniq
end

def second_text_fingerprints
  @second_text_fingerprints ||= second_text_hashes.uniq
end
```

Gambar 16. *Source code fingerprint*

d. *Similarity Fingerprint*

Berikut ini *source code* yang berfungsi untuk melakukan proses *Similarity Fingerprint* :

```
def similar_fingerprint
   first_text_fingerprints & second_text_fingerprints
  end
```

Gambar 17. *Source code similarity*

e. *Dice Coefficient*

Berikut ini *source code* yang berfungsi untuk melakukan proses *Dice Coefficient*:

```
def coeffision_similarity
   ((2.0 * similar_fingerprint.size.to_f) / ((first_text_fingerprints.size.
to_f + second_text_fingerprints.size.to_f))) * 100.0
  end
```

Gambar 18. *Source code dice coefficient*

2. Aho-Corasick

a. *N-gram*

Berikut ini *source code* yang berfungsi untuk mendapatkan nilai *N-gram* :

```
def initialize args = {}
  @first_text ||= args[:first_text]
  @second_text ||= args[:second_text]
  @ngram ||= args[:ngram] || 2
end
def first_text_ngram
  @first_text_ngram ||= first_text.to_ngram(ngram)
end
def second_text_ngram
  @second_text_ngram ||= second_text.to_ngram(ngram)
end
```

Gambar 19. Source code *N-gram*

b. Match Pattern

Berikut ini *source code* yang berfungsi untuk melakukan proses *match pattern* :

```
def first_text_match_pattern
  aho_corasick = AhoCorasick.new second_text_ngram
  matchers = aho_corasick.match(first_text)
  return matchers
end

def second_text_match_pattern
  aho_corasick = AhoCorasick.new first_text_ngram
  matchers = aho_corasick.match(second_text)
  return matchers
end

def similar_matching_pattern
  first_text_match_pattern & second_text_match_pattern
end
```

Gambar 20. Source code match pattern

c. Dice Coefficient

Berikut ini *source code* yang berfungsi untuk melakukan proses *Dice Coefficient*:

```
def coeffision_similarity
   ((2.0 * similar_matching_pattern.size.to_f) / ((first_text_ngram.
size.to_f + second_text_ngram.size.to_f))) * 100.0
end
```

Gambar 21. Source code dice coefficient

Pengujian Sistem

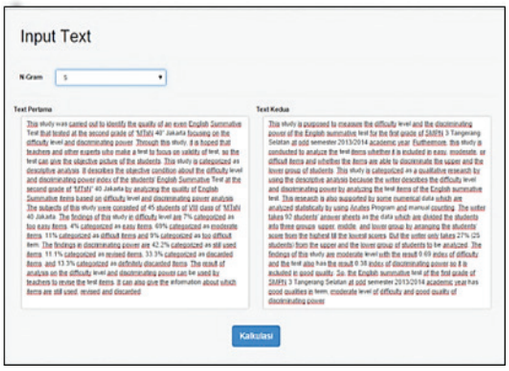
1. Hasil uji kemiripan

Data yang akan dibandingkan berupa data abstrak yang sudah ditentukan dan dipilih peneliti, berupa abstrak dalam bahasa Indonesia, Arab, dan Inggris.

No		Judul Karya Ilmiah	Hyperlink
1	a.	An item analysis of the difficulty level and discriminating power of an english summative test (a case study at the second grade of MTsN 40 Jakarta) [2013]	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=111939&lokasi=lokal
	b.	An item analysis of english summative test on difficulty level and discriminating power (a case study of the first grade students of 3 State Junior High School of Tangerang Selatan) [2014]	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=124381&lokasi=lokal
2	a.	Pengaruh kesadaran, persepsi dan preferensi konsumen terhadap perilaku konsumen dalam mengkonsumsi buah lokal: studi kasus kawasan industri di Jakarta Utara (November 2014]	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=125495&lokasi=lokal
	b.	Pengaruh kesadaran, persepsi dan preferensi konsumen terhadap perilaku mengkonsumsi buah lokal segar di kawasan perkantoran Jakarta Pusat [Desember 2014]	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=125017&lokasi=lokal
3	a.	Al-'Alaqah baina istikhdam al-Qashash al-mushowwaroh wa tanmiyah maharah al-qira'ah lada talamidz al-shaf al-tsani min al-Madrasah al-Mutawassithah al-islamiyah bi Ma'had Ulum al-Qur'an Bojongsari Depok	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=109874&lokasi=lokal
	b.	**Al-'alaqah baina al-saitharah 'ala al-mufradat fi maharah al-istima': Dirasah halah lada talamidz madrasah nur al-salam al-mutawasithah al-islamiyyah Pondok** Pinang Jakarta	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=111811&lokasi=lokal

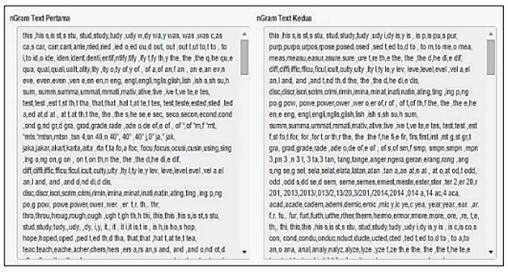
Gambar 22. Proses komparasi

Proses komparasi dengan memasukkan teks abstrak yang sudah ditentukan judulnya ke prototipe aplikasi kemiripan pola teks sebagai berikut :



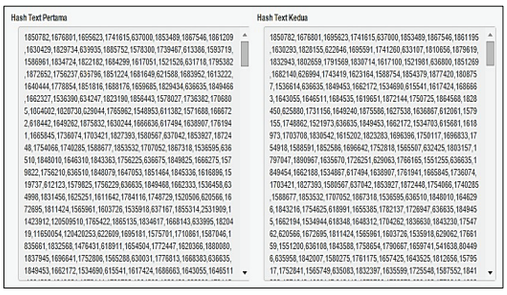
Gambar 23. Proses komparasi

Abstrak 1a dimasukkan ke teks pertama dan abstrak 1b dimasukkan ke teks kedua, kemudian tentukan nilai *N-gram* sebesar lima, selanjutnya klik tombol kalkulasi dan sistem akan memproses masing-masing teks dengan menggunakan algoritma *rabin karp* dan *aho-corasick.*



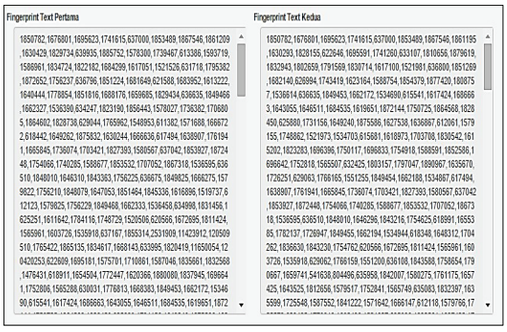
Gambar 24. *N-gram* teks pertama dan kedua *rabin karp*

Sistem akan merubah huruf kapital menjadi huruf kecil, membagi atau mengkelompokan teks sebanyak n yang dalam hal ini sebanyak lima. Kemudian dilakukan proses *hash*ing dengan fungsi rolling *hash* dari gram yang terbentuk antara teks pertama dan teks kedua akan dijelaskan sebagai berikut :

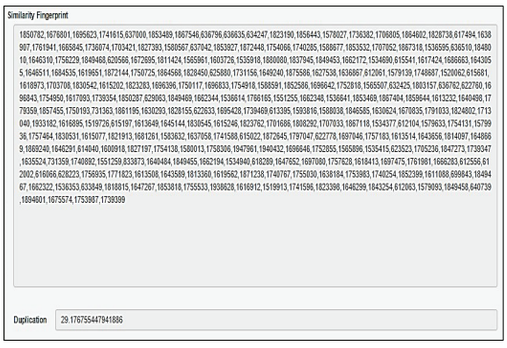


Gambar 25. *Hash* teks pertama dan kedua

Proses transformasi dari teks menjadi numerik atau angka dari serangkaian teks yang sudah dikelompokkan sebelumnya, kemudian sistem akan mencari *fingerprint* teks pertama dan teks kedua, akan dijelaskan sebagai berikut :



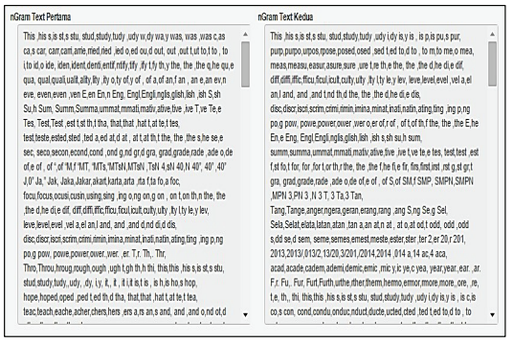
Gambar 26. *Fingerprint* teks pertama dan kedua



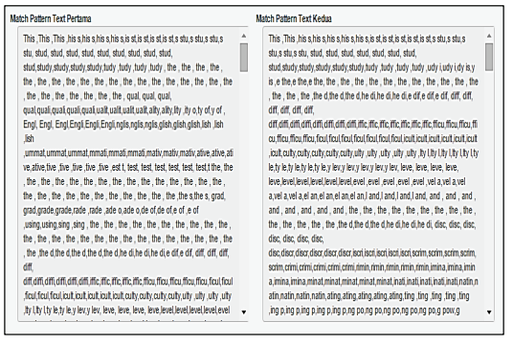
Gambar 27. *Similarity fingerprint*

Sistem akan mencari nilai yang sama antara dua teks dan menentukan persamaan kedua teks dengan hasil persamaan *dice's coefficient* sebesar 29.18%.

Proses kalkulasi algoritma *aho-corasick* akan dijelaskan sebagai berikut :

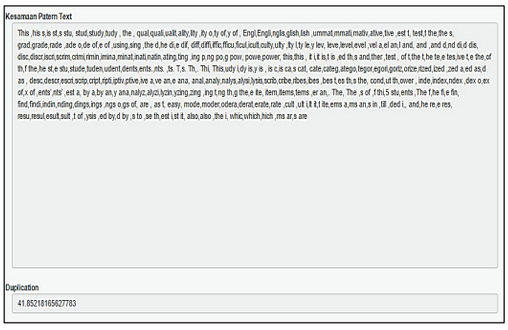


Gambar 28. *N-gram* teks pertama dan kedua *aho-corasick*



Gambar 29. *Match pattern* teks pertama dan kedua

Sistem akan membagi teks ke dalam gram-gram yang sudah ditentukan nilai *n-gram*nya.



Gambar 30. Kesamaan *pattern* teks

Sistem akan mencari kesamaan pola teks antara pola teks pertama dan teks kedua, kemudian menentukan hasil persamaan dari kedua teks dengan hasil persamaan *dice's coefficicent* sebesar 41.85%.
Sistem menampilkan *running time* kedua algoritma sebesar 0.009228377 detik untuk rabin karp dan 0.274640387 detik untuk *aho-corasick*.

Gambar 31. *Running time*

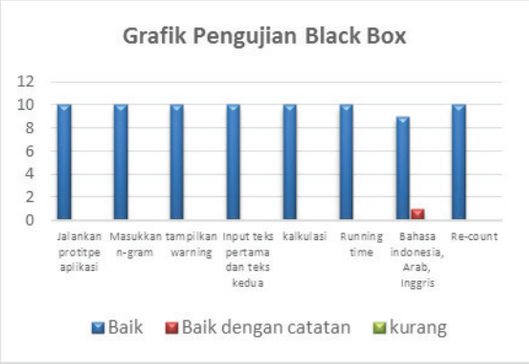
2. *Black box testing*

Pengujian *black box* dilakukan untuk memperlihatkan bahwa fungsi-fungsi perangkat lunak adalah operasional, bahwa input diterima dengan baik dan output dihasilkan dengan tepat sebagai berikut:

Tabel 1. Pengujian *Black Box*

No	*Description*	*Expected Result*	*Actual Result (%)*
1.	Menjalankan prototipe aplikasi kemiripan pola teks	Prototipe aplikasi kemiripan pola teks dapat diakses	100% OK
2.	Masukkan *N-gram*	*N-gram* dapat dipilih	100% OK
3.	Aplikasi menampilkan warning	Sistem menolak melanjutkan proses perhitungan, persyaratan teks pertama dan kedua tidak boleh kosong harus terpenuhi.	100% OK
4.	Input teks pertama dan teks kedua	Teks pertama dan teks kedua berhasil diinput	100% OK
5.	Tombol kalkulasi	Sistem memproses perhitungan masing-masing algoritma	100% OK
6.	*Running time*	Sistem dapat menampilkan waktu proses yang dihasilkan oleh algoritma Rabin Karp dan Aho-Corasick	100% OK
7.	Karya ilmiah mahasiswa dalam bahasa Indonesia, arab, inggris	Prototipe aplikasi kemiripan pola teks dapat menerapkan algoritma Rabin karp dan aho-corasick untuk menampilkan hasil perhitungan.	95% OK
8.	*Re-count*	Sistem dapat kembali ke menu awal	100% OK

Pengujian yang dilakukan melibatkan pustakawan dan staff perpustakaan yang terdiri dari, empat orang bagian IT dan Otomasi, empat orang bagian pengolahan dan dua orang bagian pengadaan. Berikut grafik pengujian *black box* :



Gambar 32. Grafik pengujian *Black box*

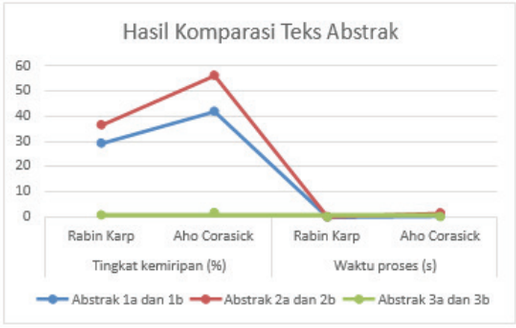
HASIL PENELITIAN

Berdasarkan hasil pengujian terhadap teks abstrak dalam bahasa Indonesia, arab dan inggris pada tabel IV.12 hasil komparasi menggunakan prototipe aplikasi *karp-corasick* diuraikan pada tabel berikut :

Tabel 2. Hasil komparasi abstrak

Teks abstrak	Tingkat kemiripan (%)		Waktu proses (s)	
	Rabin Karp	Aho Corasick	Rabin Karp	Aho Corasick
Abstrak 1a dan 1b	29.18	41.85	0.00922837	0.274640387
Abstrak 2a dan 2b	36.55	56.05	0.021360142	1.60434688
Abstrak 3a dan 3b	0.77	1.71	0.00140587	0.023114938

Hasil komparasi menunjukan bahwa, algoritma rabin karp memiliki persentase nilai lebih rendah dan waktu proses lebih cepat daripada algoritma *aho-corasick*, Penjelasan dalam bentuk lain atau menggunakan grafik sebagai berikut :



Gambar 33. Hasil komparasi teks abstrak

A. Bahasa Indonesia

Tabel 3. Hasil Pengujian Bahasa Indonesia

No		Judul Karya Ilmiah	Hyperlink
1	a.	Pengaruh kesadaran, persepsi dan preferensi konsumen terhadap perilaku konsumen dalam mengkonsumsi buah lokal: studi kasus kawasan industri di Jakarta Utara (November 2014]	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=125495&lokasi=lokal
	b.	Pengaruh kesadaran, persepsi dan preferensi konsumen terhadap perilaku mengkonsumsi buah lokal segar di kawasan perkantoran Jakarta Pusat [Desember 2014]	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=125017&lokasi=lokal
2	a.	Implementasi algoritma vigenere cipher untuk aplikasi enkripsi sms berbasis android (2012)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=110395&lokasi=lokal
	b.	Aplikasi SMS terenkripsi di android dengan caesar cipher dan kombinasi algoritma vigenere (2013)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=120910&lokasi=lokal
3	a.	Aplikasi chatting menggunakan kriptografi enkripsi blowfish (2011)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=104658&lokasi=lokal
	b.	Aplikasi instant messaging menggunakan enkripsi blowfish (2014)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=120875&lokasi=lokal
4	a.	Pengaruh pemahaman, kesadaran dan persepsi wajib pajak Bumi dan bangunan terhadap keberhasilan penerimaan pajak Bumi dan Bangunan (2010)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=96165&lokasi=lokal
	b.	Pengaruh Jumlah Wajib Pajak, Perilaku Wajib Pajak, dan Pemeriksaan Pajak terhadap Penerimaan Pajak dengan Kesadaran Wajib Pajak sebagai Variabel Moderating: Studi Kasus di KPP Pratama Wilayah DKI Jakarta (2013)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=108944&lokasi=lokal
5	a.	Analisis Pengaruh Kualitas Pelayanan, Citra Perusahaan, dan Kepuasan Konsumen terhadap Loyalitas Konsumen (2011)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=108945&lokasi=lokal
	b.	Analisis pengaruh kualitas pelayanan dan harga terhadap kepuasan konsumen dan dampaknya terhadap loyalitas konsumen (Studi Kasus Pada Pasien Rumah Sakit Mulya) (2014)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=119525&lokasi=lokal
6	a.	Pengaruh brand image, kepuasan konsumen dan faktor demografi terhadap loyalitas konsumen (2013)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=112412&lokasi=lokal
	b.	Pengaruh persepsi kualitas produk dan brand image terhadap loyalitas konsumen (2014)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=120327&lokasi=lokal
7	a.	Analisis Pengaruh Kinerja Pelayanan Dan Kepuasan Terhadap Loyalitas Pelanggan : Studi Kasus pada PT.SUN Beach Indonesia Jakarta (2008)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=83680&lokasi=lokal
	b.	Pengaruh kualitas pelayanan terhadap loyalitas pelanggan pada PT. Garuda Indonesia (2013)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=116118&lokasi=lokal
8	a.	Pengaruh kompensasi, motivasi, komitmen organisasional dan kepemimpinan terhadap kinerja karyawan bagian akuntansi (2013)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=116824&lokasi=lokal
	b.	Pengaruh kompensasi motivasi dan komitmen organisasi terhadap kinerja marketing pada asuransi syariah (2014)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=123359&lokasi=lokal
9	a.	Pengaruh gaya kepemimpinan, disiplin kerja, kompetensi dan komitmen organisasi terhadap kinerja karyawan PT. Takenaka Indonesia (2013)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=117328&lokasi=lokal
	b.	Pengaruh kompensasi, gaya kepemimpinan, disiplin kerja dan karakteristik individu terhadap kinerja karyawan PT. Vidya Rejeki Tama (2014)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=124225&lokasi=lokal
10	a.	Pengaruh model pembelajaran reciprocal teaching (pengajaran berbalik) terhadap hasil belajar Biologi siswa pada konsep protista (eksperimen di MAN 2 Bogor) (2010)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=92155&lokasi=lokal
	b.	Penerapan model pembelajaran terbalik reciprocal teaching untuk meningkatkan aktivitas belajar matematika siswa : penelitian tindakan kelas di mts daarul hikmah pamulang (2010)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=93822&lokasi=lokal

Kemudian dilakukan pengujian kembali terhadap 10 (sepuluh) teks abstrak dalam bahasa Indonesia, Arab, dan Inggris.

Tabel 4. Nilai presentase teks Bahasa Indonesia

Teks abstrak	Tingkat kemiripan (%)		Waktu proses (s)	
	Rabin Karp	Aho Corasick	Rabin Karp	Aho Corasick
Abstrak 1a dan 1b	36.56	56.08	0.022868258	1.772802362
Abstrak 2a dan 2b	25.91	46.65	0.008520567	0.199963903
Abstrak 3a dan 3b	27.71	39.22	0.008009062	0.196887192
Abstrak 4a dan 4b	37.01	54.96	0.006253455	0.184001668
Abstrak 5a dan 5b	34.12	65.61	0.005833158	0.142244721
Abstrak 6a dan 6b	31.89	46.61	0.011583614	0.579011153
Abstrak 7a dan 7b	27.14	31.06	0.010734879	0.316052974
Abstrak 8a dan 8b	27.13	31.62	0.006800175	0.134687982
Abstrak 9a dan 9b	56.65	62.73	0.00651835	0.16769145
Abstrak 10a dan 10b	28.27	50.57	0.006900656	0.15743526
Average	**33.239**	**48.511**		

B. Bahasa Arab

Tabel 5. Hasil Pengujian Bahasa Arab

No		Judul Karya Ilmiah	Hyperlink
1	a.	تحليل بند من مستوى الصعوبة والقوة تميز لاختبار تلخيصي الانجليزية	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=111939&lokasi=lokal
	b.	تحليل بنود الاختبار التحصيلي الانجليزية على مستوى الصعوبة والقوة تميز	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=124381&lokasi=lokal
2	a.	تحليل لمستوى المقروئية من نصوص القراءة في جواز السفر للعالم هما كتاب باستخدام انهيار اختبار	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=111177&lokasi=lokal
	b.	تحليل لمستوى المقروئية من النصوص قراءة في كتاب الفعالة الإنجليزية	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=119406&lokasi=lokal
3	a.	تأثير التعويض، والدافع، والالتزام التنظيمي والقيادة على أداء الموظفين المحاسبة	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=116824&lokasi=lokal
	b.	تأثير الدافع التعويض والالتزام الأداء التسويقي للمنظمة على التكافل	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=123359&lokasi=lokal
4	a.	تحليل خطأ على طلاب الصف الثالث الكتابة السردية	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=94561&lokasi=lokal
	b.	تحليل الطلاب الأخطاء النحوية في كتابة النص السردي: دراسة حالة في الصف الثاني	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=111248&lokasi=lokal
5	a.	تحليل على الطلاب أخطاء في الكتابة الوصفية: دراسة حالة في الصف الأول	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=112754&lokasi=lokal
	b.	تحليل خطأ على الطلاب كتابة نص وصفي دراسة حالة في طلاب الصف الثاني	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=123722&lokasi=lokal
6	a.	تحليل على أخطاء الطلاب في استخدام ضمير شخصي دراسة حالة الطلاب في الصف الأول من المدرسة الثانوية	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=120502&lokasi=lokal
	b.	تحليل على أخطاء الطلاب في استخدام الضمائر الشخصية دراسة حالة في الصف الأول من المدرسة الثانوية	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=122455&lokasi=lokal
7	a.	تحليل على أخطاء الطلاب في استخدام الماضي التام المستمر متوترة	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=109913&lokasi=lokal
	b.	تحليل على أخطاء الطلاب في استخدام المضارع التام المستمر متوترة	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=119012&lokasi=lokal

No		Judul Karya Ilmiah	Hyperlink
8	a.	تحليل مواد علم السامية على المرأة الإعلانات العطر في مجلة كوزموبوليتان	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=80587&lokasi=lokal
	b.	تحليل اللغة المجازية في خطوط العلامة الإعلان في مجلة تاتلر إندونيسيا	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=120015&lokasi=lokal
9	a.	طلاب تحسين وصفي الكتابة من خلال رسم خرائط العقل أحد الفصول البحث العملي في السنة الثانية	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=112533&lokasi=lokal
	b.	تطبيق استراتيجية رسم خرائط العقل لتحسين قدرة الطلاب الكتابة في نص وصفي: بحث إجراءات الفصول الدراسية في الصف الثاني	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=125698&lokasi=lokal
10	a.	تحسين الطلاب القراءة والفهم من النص السردي من خلال رسم الخرائط قصة	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=120373&lokasi=lokal
	b.	تحسين الطلاب القراءة والفهم من النص السردي من خلال رسم الخرائط القصة: بحث إجراءات الفصول الدراسية في ثمانية طلاب الصف	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=125257&lokasi=lokal

Tabel 6. Nilai presentase teks Bahasa Arab

Teks abstrak	Tingkat kemiripan (%)		Waktu proses (s)	
	Rabin Karp	Aho Corasick	Rabin Karp	Aho Corasick
Abstrak 1a dan 1b	26.55	38.97	0.010439491	0.257933665
Abstrak 2a dan 2b	22.78	35.25	0.01361233	0.374345318
Abstrak 3a dan 3b	18.87	22.26	0.01111799	0.119248616
Abstrak 4a dan 4b	19.99	27.64	0.028275104	0.206127278
Abstrak 5a dan 5b	19.67	32.66	0.006411684	0.083065101
Abstrak 6a dan 6b	22.93	32.46	0.007369917	0.104890028
Abstrak 7a dan 7b	18.65	28.27	0.012795121	0.192311343
Abstrak 8a dan 8b	9.06	13.24	0.00969558	0.130361871
Abstrak 9a dan 9b	39.58	55.16	0.012461218	0.320976744
Abstrak 10a dan 10b	37.84	50.91	0.009376853	0.194788583
Average	**23.592**	**33.682**		

C. Bahasa Inggris

Tabel 7. Hasil Pengujian Bahasa Inggris

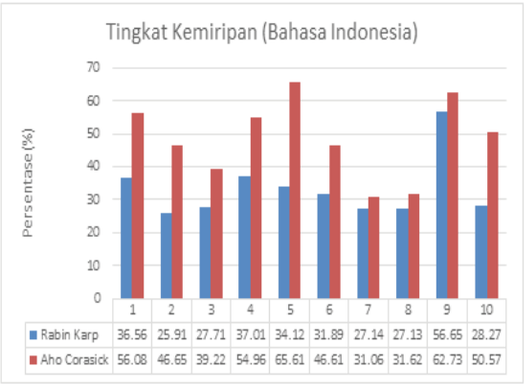
No		Judul Karya Ilmiah	Hyperlink
1	a.	An item analysis of the difficulty level and discriminating power of an english summative test (a case study at the second grade of MTsN 40 Jakarta) [2013]	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=111939&lokasi=lokal
	b.	An item analysis of english summative test on difficulty level and discriminating power (a case study of the first grade students of 3 *State* Junior High School of Tangerang Selatan) [2014]	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=124381&lokasi=lokal
2	a.	An Analysis of The Readability Level of Reading Texts in Passport to The World 2 Textbook by Using Cloze Test: A Case Study at the Eighth Grade Students of SMP 3 Tangerang Selatan [2013]	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=111177&lokasi=lokal
	b.	An Analysis of the Readability Level of the Reading Texts in the Effective English Textbook; A Case Study at the Eighth Grade Students of MTs. Al-Falah, Jakarta Selatan [2014]	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=119406&lokasi=lokal
3	a.	The analysis of english textbook english on SKY 2 Esed at the second year of SMP Dharma Karya, Pamulang [2010]	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=95862&lokasi=lokal
	b.	Genre analysis of the reading texts in english on sky 2 textbook for the second grade of junior secondary school at SMPN 10 Ciputat [2014]	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=120815&lokasi=lokal
4	a.	Error analysis on the third grade students narrative writing at SMA Mandiri Balaraja (2009)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=94561&lokasi=lokal
	b.	Analysis of the students grammatical errors in narative text writing: A case study at the second grade of SMPN 2 Tangerang Selatan (2013)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=111248&lokasi=lokal

No		Judul Karya Ilmiah	Hyperlink
5	a.	An analysis on students errors in descriptive writing: A case study at the first grade of SMA Negeri 37 Jakarta (2013)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=112754&lokasi=lokal
	b.	Error analysis on the students writing of descriptive text (a case study at second grade students of SMP PGRI 2 Ciputat) (2014)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=123722&lokasi=lokal
6	a.	An analysis on students errors in using personal pronoun (a case study of students on first grade of SMP Islam Al-Syukro, Ciputat) (2014)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=120502&lokasi=lokal
	b.	An analysis on students errors in using personal pronouns (a case study at the first grade of junior high school of Yayasan Miftahul Jannah) (2014)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=122455&lokasi=lokal
7	a.	An analysis on students errors in using past perfect continuous tense (2012)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=109913&lokasi=lokal
	b.	An analysis on students errors in using present perfect continuous tense (A Case Study at First Grade Students of SMAN 63 Jakarta) (2014)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=119012&lokasi=lokal
8	a.	A Semiotic Analysis On Women Fragrance Advertisements In Cosmopolitan Magazine (2009)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=80587&lokasi=lokal
	b.	Analysis of figurative language in the advertisement taglines in Indonesia tatler magazine (2014)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=120015&lokasi=lokal
9	a.	Improving students descriptive writing through mind mapping (A Classroom Action Research in the Second Year of Bina Prestasi 3 Class of MTsN Tangerang Pamulang) (2010)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=112533&lokasi=lokal
	b.	Applying mind mapping strategy to improve students writing ability in descriptive text: a classroom action research at the second grade of SMP Al-Mizan Pandeglang-Banten (2014)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=125698&lokasi=lokal
10	a.	Improving the students reading comprehension of narrative text through story mapping (a classroom action research of the second grade of MTs Tarbiyatul Falah) (2014)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=120373&lokasi=lokal
	b.	Improving students reading comprehension of narrative text through story mapping: a classroom action research at eight grade student of SMP PGRI 2 Ciputat (2014)	http://tulis.uinjkt.ac.id/opac/themes/katalog/detail.jsp?id=125257&lokasi=lokal

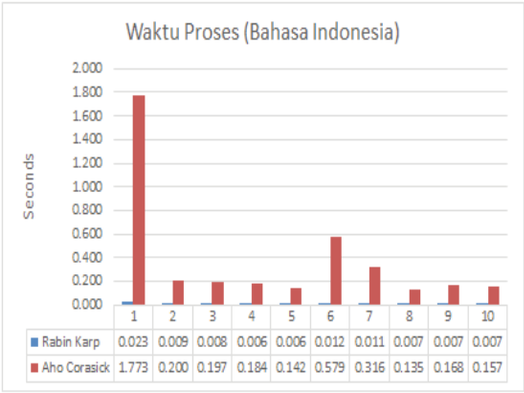
Tabel 8. Nilai presentase teks Bahasa Iggris

Teks abstrak	Tingkat kemiripan (%)		Waktu proses (s)	
	Rabin Karp	Aho Corasick	Rabin Karp	Aho Corasick
Abstrak 1a dan 1b	29.17	41.85	0.010042574	0.319871235
Abstrak 2a dan 2b	34.92	48.19	0.02020552	0.915169001
Abstrak 3a dan 3b	29.13	40.13	0.013207198	0.359362437
Abstrak 4a dan 4b	23.77	30.86	0.030393627	0.593820494
Abstrak 5a dan 5b	26.99	41.59	0.00600003	0.163059389
Abstrak 6a dan 6b	27.81	37.72	0.010242535	0.273385519
Abstrak 7a dan 7b	27.61	36.79	0.014457778	0.321859765
Abstrak 8a dan 8b	14.75	20.41	0.015865208	0.209340493
Abstrak 9a dan 9b	48.99	63.43	0.012247514	0.526337282
Abstrak 10a dan 10b	47.44	59.98	0.024177125	0.323686232
Average	**31.058**	**42.095**		

Hasil yang didapatkan yaitu, tingkat kemiripan atau persentase (dalam bahasa indonesia) algoritma *rabin karp* lebih rendah dengan rata-rata 33,23% dengan nilai minimum *running time* 0.005833158 detik dan nilai maksimum *running time* 0.022868258 detik. sedangkan algoritma *aho-corasick* rata-rata 48,51% dengan nilai minimum *running time* 0.134687982 detik dan nilai maksimum *running time* 1.772802362 detik.

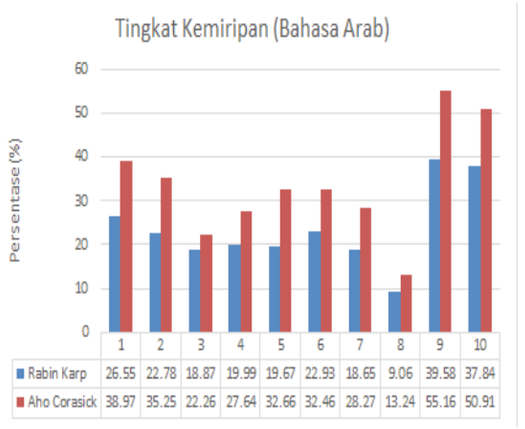


Gambar 34. Grafik pengujian persentase

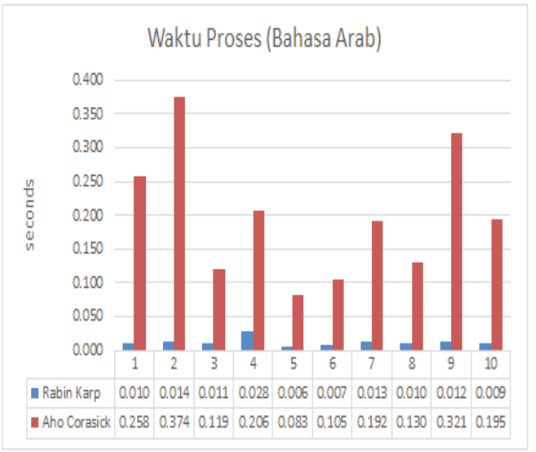


Gambar 35. Grafik pengujian waktu proses

Tingkat kemiripan atau persentase (dalam bahasa Arab) algoritma *rabin karp* lebih rendah dengan rata-rata 23,59% dengan nilai minimum *running time* 0.006411684 detik dan nilai maksimum *running time* 0.028275104 detik. sedangkan algoritma *aho-corasick* rata-rata 33,68% dengan nilai minimum *running time* 0.083065101 detik dan nilai maksimum *running time* 0.374345318 detik

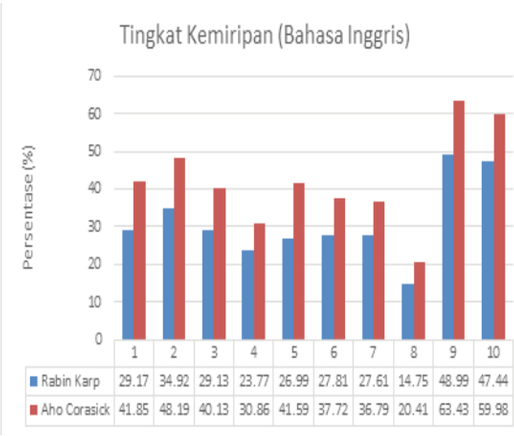


Gambar 36. Grafik pengujian persentase

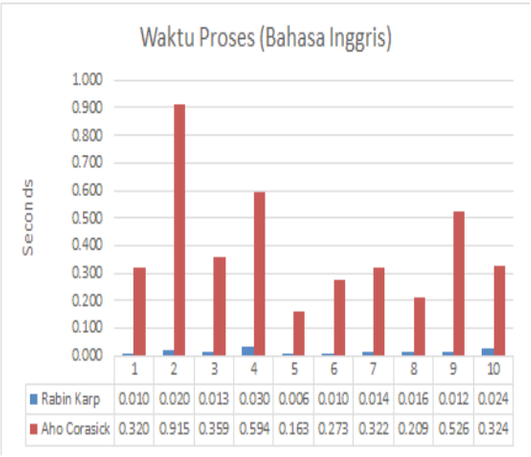


Gambar 37. Grafik pengujian waktu proses

Tingkat kemiripan atau persentase (dalam bahasa Inggris) algoritma *rabin karp* lebih rendah dengan rata-rata 31,05% dengan nilai minimum *running time* 0.00600003 detik dan nilai maksimum *running time* 0.030393627 detik. sedangkan algoritma *aho-corasick* rata-rata 42,09% dengan nilai minimum *running time* 0.163059389 detik dan nilai maksimum *running time* 0.915169001 detik.



Gambar 38. Grafik pengujian persentase



Gambar 39. Grafik pengujian waktu proses

Berdasarkan beberapa hasil pengujian serta analisis yang telah dilakukan hasil komparasi menunjukan bahwa, algoritma *rabin karp* memiliki persentase nilai lebih rendah dan waktu proses lebih cepat daripada algoritma *aho-corasick*.

PENUTUP

Kesimpulan

Perbandingan algortima *string matching* antara *Rabin Karp* dengan *Aho-Corasick* pada abstrak karya ilmiah mahasiswa menghasilkan output akurasi (*percentage similarity)* dan waktu proses (*time complexity*) kemiripan pola teks yang dilakukan algoritma *rabin karp* lebih baik dari algoritma *aho-corasick*.

Pengujian yang dilakukan menunjukkan prototipe aplikasi kemiripan pola teks mampu melakukan komparasi dan memproses algoritma *string matching* antara algoritma *rabin karp* dan *aho-corasick* dengan menggunakan bahasa indonesia, inggris, dan arab.

Saran

Berdasarkan hasil penelitian yang dilakukan perlu pengembangan riset lebih lanjut demi kesempurnaan *prototype* aplikasi yang memberikan kemudahan *user* terhadap data yang akan di cek kemiripannya (*similarity*). Oleh karena itu saran yang dapat peneliti berikan yaitu :

1. Komparasi tidak hanya melalui teks abstrak tetapi dapat mendukung full teks digital dalam berbagai format (.pdf, .docx, .rtf, .txt, dan format digital lainnya).
2. Untuk melihat tingkat akurasi dari algoritma, akan lebih baik lagi bila dikomparasi dengan model algoritma lain.
3. Integrasi dengan database sistem perpustakaan.

DAFTAR PUSTAKA

Lusia Kus Anna, "Dugaan Plagiat PTN diselidiki", 2012, http://edukasi.kompas.com/read/2012/06/06/09355010/Dugaan.Plagiat.di.PTN.Diselidiki.

Munir, Rinaldi. *Algoritma dan Pemrograman dalam Bahasa Pascal dan C*, Buku 1. Bandung:Informatika, 2001.

Suarga, *Algoritma dan Pemrograman*. yogyakarta:ANDI, 2004.

Syaroni, Mokhamad dan Rinaldi Munir., Pencocokan *String* Berdasarkan Kemiripan Ucapan (Phonetic *String Matching*) dalam Bahasa Inggris. Bandung: Institut Teknologi Bandung, 2004.

Arliadinda. "Pencocokan *String* dengan Menggunakan Algoritma Rabin-Karp dan Shift-Or". Bandung:Sekolah Tinggi Teknologi Telkom, 2005.

Atmopawiro. "Pengkajian dan Analisa Tiga Algoritma Efisien Rabin-Karp, Knuth Morris Prat, dan Aho-Corasick Dalam Pencarian Pola Pada Suatu Teks", Bandung : Institut Teknologi Bandung

Wirawan, Teddy Pandu. *Penggunaan Algoritma Rabin-Karp Dalam Pencocokan String*, Bandung : Institut Teknologi Bandung, 2008.

Baedlowi, Nadjib dan Deka Aditia Adam. *String Matching* Menggunakan Algoritma Rabin-Karp, Bandung:Institut Teknologi Bandung, 2006.

Aho, A.V. & Corasick, M. J., Efficient *String Matching*: An Aid to Bibliographic Search. Communications Of The ACM. Volume 18, Issue 6, p333-340, 1975.

Handoko, Andrew. "File Undelete Untuk Memulihkan File Yang Telah Terhapus Dari File System Dengan Algoritma Aho-Corasick". Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara, 2014.

Vilo, J. *Multiple pattern matching*, 2008.

Lenz, Patrick. *Simply Rails 2*. SitePoint Pty. Ltd. Cambrigde, 2008.

Ruby, Sam., Thomas, Dave., Heinemeier, David Hanson. "Agile Web Development with *Rails* (Fourth Edition)". The Pragmatic Programers LLC, 2010. *Getting Started with Rails.* http://guides.rubyon*rails*.org/, 2017.

Agustinus, N. "Studi Analisis *Rapid Application Development* Sebagai Salah Satu Alternatif Metode Pengembangan Perangkat Lunak". Jurnal Informatika, Vol.3, 2002. Hal 74-79. Universitas Kristen Petra, 2002.

Mc.,Leod, R. Jr. *System Development: A Project Management Approach*. New York: Leigh Publishing LLC, 2002.

Bentley, J. & Sedgewick, R. Ternary Search Trees. Dr. Dobb's Journal, 1998.

Kendall, J.E. & Kendall, K.E, Analisis dan Perancangan Sistem. Jakarta:Indeks, 2010.

Moedjiono, *Pedoman Penelitian, Penyusunan dan Penilaian Tesis (V.5),* Pascasarjana Universitas Budi Luhur, 2012.

Kusmana, Suherli. *Merancang Karya Tulis Ilmiah,* Bandung:PT Remaja Rosdakarya, 2010.

Pressman, Roger S.. *Rekayasa Perangkat Lunak: pendekatan praktisi (Buku I)*. Yogyakarta:ANDI, 2002.

Kendall, J.E. & Kendall, K.E., Analisis dan Perancangan Sistem, Edisi 5, Jilid 1., Jakarta:Indeks, 2008.

Munawar. *Pemodelan Visual dengan UML*, Graha Ilmu: Yogyakarta, 2005.

Pandawa, Jesry Prasasty. "Perancangan Sistem Deteksi Kemiripan Dokumen Menggunakan Algoritma Winnowing", Karya Ilmiah Fakultas Sains dan Teknologi UIN Syarif Hidayatullah Jakarta, 2014.

Ersin, Abdul Kadir, et al. "The Efficiency of *String Matching* Algorithm on Natural Language Teks", International Scientific Conference of Unitech, PP. 349-352, Gabrovo, 2007.

Kurniawati, Ana dan I Wayan Simri Wicaksana, Perbandingan Pendekatan Deteksi Plagiarism Dokumen Dalam Bahasa Inggris, Proceeding, Seminar Ilmiah Nasional Komputer dan Sistem Intelijen (KOMMIT 2008), ISSN: 1411-6286, PP. 284-291, 2008.