Pemanfaatan Algoritma Pattern Matching dan Regex Dalam Pencarian Informasi Pada Dokumen Karya Ilmiah

Ahmad Rafi Maliki - 13522137[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung,
Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522137@std.stei.itb.ac.id

***Abstract*—Seiring dengan berkembangnya ilmu sains, jumlah dokumen karya ilmiah yang dipublikasikan juga meningkat. Perkembangan Intelegensi Buatan dalam beberapa tahun terakhir diharapkan dapat mempercepat kemajuan ilmu sains. Untuk itu, diperlukan sebuah intelegensi buatan yang mampu membaca dan menganalisis dokumen karya ilmiah yang dipublikasikan. Penelitian ini bertujuan untuk menciptakan landasan bagi algoritma yang akan digunakan dalam *machine learning* untuk menganalisis dokumen karya ilmiah. Dengan menggunakan pendekatan algoritma string matching dan regex, penelitian ini berupaya mengidentifikasi informasi penting yang terdapat dalam sebuah dokumen karya ilmiah. Fokus penelitian ini adalah menentukan topik-topik yang paling sering dan penting dibahas dalam karya ilmiah. Diharapkan, hasil penelitian ini akan menjadi landasan yang kuat untuk pengembangan algoritma *machine learning* yang lebih canggih di masa mendatang.**

***Kata kunci* — Sains, Dokumen Karya Ilmiah, *String Matching,* Regex, Algoritma.**

I. Pendahuluan

Kemajuan ilmu pengetahuan merupakan dambaan seluruh akademisi di dunia. Salah satu indikator berkembangnya ilmu pengetahuan adalah banyaknya dokumen karya ilmiah yang dipublikasikan. Publikasi dokumen ini memungkinkan akademisi di seluruh dunia untuk bertukar informasi mengenai temuan-temuan baru, sehingga mempercepat perkembangan ilmu pengetahuan.

Dalam beberapa tahun terakhir, artificial intelligence (AI) chatbot seperti ChatGPT yang dikembangkan oleh OpenAI atau Copilot yang dikembangkan oleh Microsoft, menjadi sangat populer. Chatbot semacam ini banyak digunakan karena pengguna merasa seolah berkomunikasi dengan orang lain dan dapat mengajukan pertanyaan menggunakan bahasa sehari-hari dengan mudah. Bahkan, penggunaan chatbot AI kini mulai menggeser pencarian melalui mesin pencari seperti Google.

Melihat perkembangan ini, muncul ide bahwa chatbot AI dapat menggunakan dokumen karya ilmiah sebagai model pelatihan untuk memperkaya pengetahuan mereka. Namun, tantangannya adalah bagaimana program dapat menganalisis sebuah dokumen karya ilmiah secara efektif.

Berdasarkan pembelajaran dari mata kuliah Strategi Algoritma, penulis memiliki ide untuk mengembangkan algoritma yang dapat mengidentifikasi topik yang dibahas dalam sebuah karya ilmiah, sehingga mempermudah analisis lebih lanjut menggunakan metode statistik atau algoritma machine learning yang lebih canggih. Dengan menggunakan kombinasi algoritma string matching dan regex, kita dapat mengetahui topik-topik yang paling sering disebutkan dan dibahas dalam sebuah dokumen karya ilmiah. Namun, karena keterbatasan algoritma ini, analisis hanya dapat dilakukan pada teks dan bukan gambar.

Penelitian ini diharapkan dapat menjadi landasan bagi penelitian selanjutnya terkait dengan pengembangan analisis intelegensi buatan terhadap dokumen karya ilmiah.

II. Studi Literatur

A. Dokumen Karya Ilmiah

Dokumen Karya Ilmiah, Jurnal Karya Ilmiah, ataupun *Scientific Paper* merupakan sebuah dokumen yang ditulis para akademisi dan peneliti untuk berbagi hasil penelitiannya dengan peneliti lain untuk dianalisa atau diteliti lebih lanjut. Maka dari itu dokumen ini merupakan faktor kritikal dalam perkembangan ilmu sains, dimana hasil kerja dari seorang peneliti dapat menjadi landasan kerja peneliti lainnya.

Sebuah dokumen karya ilmiah, harus mudah untuk dibaca dengan jelas, tepat, dan akurat agar informasi yang terkandung didalamnya mampu disampaikan secara utuh tanpa ada misinterpretasi informasi. Sebuah dokumen karya ilmiah, biasanya dikutip oleh peneliti lain jika dokumen tersebut berguna dalam penelitiannya.

Sebuah dokumen ilmiah biasanya terdiri dari bagian pendahuluan yang berisi tentang alasan dibuatnya dokumen tersebut atau alasan dilakukannya penelitian terkait. Selain itu dokumen karya ilmiah juga mengandung studi literatur, metode penelitian, analisis pembahasan, serta kesimpulan dari penelitian yang telah dilakukan.

B. *Pattern Matching*

Pattern matching merupakan metode yang digunakan untuk mencari dan mengidentifikasi keberadaan urutan karakter, kata, atau pola dalam data yang lebih besar. Metode ini sangat berharga dalam berbagai aplikasi seperti pengolahan teks, analisis data, dan pengembangan perangkat lunak, karena memfasilitasi pencarian dan manipulasi informasi secara efisien.

Konsep dasar dari *pattern matching* melibatkan dua komponen utama: teks/*text* (T) yang merupakan string panjang dan pola/*pattern* (P) yang merupakan string yang lebih pendek untuk dicari dalam teks. Tujuannya adalah untuk menemukan lokasi pertama dalam teks dimana terdapat kecocokan lengkap dengan pola.

Sebagai contoh dalam teks “Saya menyukai mata kuliah Strategi Algoritma.” dan dengan pola “mata”, metode ini bertujuan untuk menemukan keberadaan “mata” pertama kali dalam teks tersebut. Metode ini memanfaatkan algoritma efisien untuk mempercepat pencarian dan mengurangi kompleksitas komputasi, terutama pada teks yang sangat panjang.

Konsep penting lainnya yang perlu dipahami dalam membahas *string* dalam konteks *pattern matching* adalah *prefix* yang merupakan *substring* (bagian *string* yang lebih kecil) yang dimulai dari indeks nol sampai indeks tertentu yang bukan akhir string dan juga *suffix* yang merupakan *substring* yang dimulai dari indeks tertentu bukan nol hingga akhir string.

Pemahaman mengenai *string, prefix,* dan *suffix* menjadi penting dalam membahas konsep algoritma *pattern matching* yang digunakan dalam penelitian ini yaitu algoritma Knuth-Morris-Pratt (KMP) dan juga Boyer-Moore (BM).

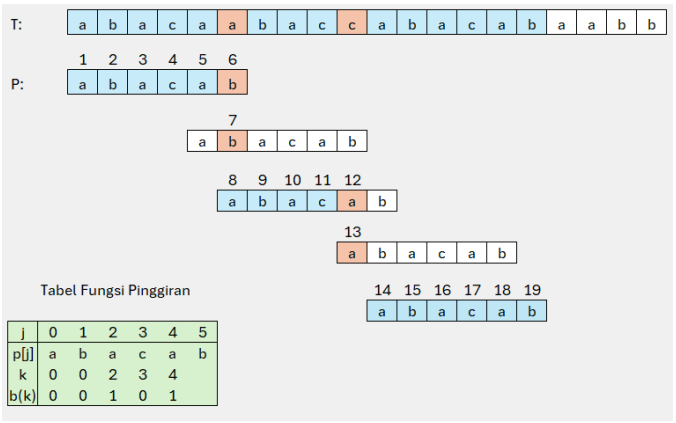
C. *Algoritma Knuth-Morris-Pratt*

Algoritma Knuth-Morris-Pratt (KMP) merupakan salah satu algoritma yang digunakan dalam *pattern matching*. Algoritma ini dikembangkan oleh James H. Morris, Donald Knuth, dan Vaughan Pratt pada tahun 1970, dan dipublikasi pada tahun 1977. Algoritma ini dirancang untuk mencari keberadaan sebuah pola dalam teks dengan cara yang lebih cerdas dibandingkan dengan pendekatan *brute force*. Algoritma KMP melakukan pencarian dengan menghindari pengulangan pencocokan yang tidak perlu dengan memanfaatkan informasi yang sudah didapatkan dari pencocokan sebelumnya. Hal ini mampu dicapai menggunakan fungsi pinggiran, yang juga dikenal sebagai fungsi kegagalan, yang menentukan seberapa jauh pola harus digeser saat terjadi ketidakcocokan.

Fungsi pinggiran (*border function*) merupakan inti dari algoritma KMP. Fungsi ini mempercepat proses pencarian dengan mengidentifikasi bagian terpanjang dari *prefix* pola yang juga merupakan *suffix*. Dengan tambahan informasi ini, KMP mampu melanjutkan pencocokan dari titik yang paling mungkin menghasilkan kesesuaian tanpa harus memulai kembali dari awal pola. Misalnya jika terjadi ketidakcocokan pada posisi j dari pola, algoritma dapat menggunakan fungsi pinggiran untuk menemukan nilai j baru sehingga meminimalisir bagian dari teks yang perlu dicocokkan kembali.

Salah satu keunggulan utama dari algoritma KMP adalah efisiensi waktu yang tinggi. Dengan melakukan *preprocessing* pada pola untuk menghitung fungsi pinggiran, pencarian string bisa dilakukan dalam waktu linier terhadap panjang teks dan pola, yaitu O(m+n), dimana m adalah panjang pola dan n adalah panjang teks. Hal tersebut mengakibatkan algoritma KMP sangat cocok untuk pencarian pada teks yang panjang.

Sayangnya, algoritma ini memiliki kelemahan jika digunakan untuk melakukan *pattern matching* pada teks yang memiliki ukuran alphabet besar, karena memungkinkan ketidakcocokan terjadi sangat dini di awal pencarian sehingga efektivitas dari fungsi pinggiran berkurang yang mana fungsi ini akan unggul apabila ketidakcocokan terjadi di akhir pola.



Gambar 1. Ilustrasi Algoritma KMP (Sumber: [1])

Secara sistematis, algoritma KMP dapat diformulasikan sebagai berikut. [1] Hitung fungsi pinggiran *b* untuk pola *s*. [2] Tetapkan *substring* pertama pada teks dengan panjang |s|. [3] Bandingkan substring tersebut dengan pola, apabila seluruh simbol sesuai simpan indeks ditemukan kecocokan. Selainnya hitung panjang *L* dimana terjadi kecocokan, apabila L≠0, geser pola sejauh *L*-*b*(*L*-1) simbol, jika *L*=0 geser pola sejauh 1 simbol. [4] Ulangi langkah 3 sampai seluruh *substring* diproses, dan kembalikan indeks ditemukan atau kembalikan -1 jika tidak terjadi kecocokan..

D. *Algoritma Boyer-Moore*

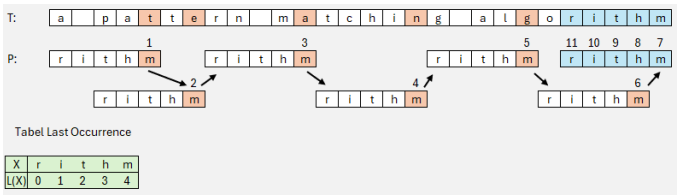
Algoritma Boyer-Moore adalah salah satu metode paling efisien untuk *pattern matching*. Algoritma ini dikembangkan oleh Robert S. Boyer dan J Strother Moore pada tahun 1977, dan mendapat popularitasnya karena memiliki kecepatan tinggi dalam aplikasi praktis ketimbang dengan algoritma *brute force* ataupun KMP. Algoritma ini merupakan improvisasi dari KMP dengan cara mengubah arah pencocokan simbol menjadi dari kanan ke kiri. Sementara pola tetap digeser ke kanan.

Jika ditemukan ketidakcocokan saat pengecekan simbol, pola akan digeser ke kanan jika mungkin dengan cara melakukan *alignment* dengan kemunculan terakhir simbol

yang cocok. Teknik ini dikenal dengan teknik cermin (*looking-glass technique*) dan teknik lompat karakter (*character-jump technique*) yang memanfaatkan tabel kemunculan terakhir atau *last occurrence table.* Teknik ini berkontribusi besar dalam mengurangi jumlah perbandingan yang diperlukan.

Teknik cermin dalam algoritma ini mengubah cara pencocokan karakter dalam pola teks. Ketimbang memulai dari karakter pertama, algoritma ini memulai pencocokan dari karakter terakhir pola. Pendekatan ini memungkinkan deteksi ketidakcocokan lebih dini, sehingga dapat dengan cepat menggeser pola tanpa perlu mengecek setiap karakter dari awal. Proses ini akan secara signifikan mengurangi waktu pengecekan pada teks yang panjang.

Teknik lompat karakter mengambil keuntungan dari informasi tentang kegagalan pencocokan karakter. Ketika terjadi ketidakcocokan, algoritma ini akan menggeser pola ke indeks tertentu sesuai dengan data yang diperoleh dari fungsi *last occurrence L(x)*. Fungsi ini memberikan posisi terakhir dari setiap karakter dalam pola. Teknik ini menyebabkan algoritma BM memiliki kompleksitas waktu O(mn+A), dimana m adalah panjang pola, n adalah panjang teks, dan A adalah ukuran alphabet.



Gambar 2. Ilustrasi Algoritma BM (Sumber: [1])

Ada tiga kasus yang dapat terjadi ketika terjadi ketidak cocokan. [1] Jika karakter yang tidak cocok pada teks (*x*) terdapat di suatu tempat dalam pola, algoritma akan mencoba menggeser pola ke kanan untuk menyelaraskan kemunculan terakhir *x* dalam pola dengan karakter ini di teks. [2] Jika *x* terdapat pada pola, tetapi pergeseran ke kanan ke kemunculan terakhir tidak memungkinkan, maka pola digeser satu karakter ke kanan. [3] Jika *x* tidak terdapat pada pola, maka pola digeser sepenuhnya untuk memulai pencocokan baru tepat setelah karakter *x* ini di teks.

E. Regex

Regex atau *Regular Expression* merupakan notasi standar yang mendeskripsikan suatu pola berupa urutan karakter atau string. Regex digunakan untuk pencocokan string (*pattern matching*) dengan efisien. Regex sudah menjadi standar yang tersebar di semua *tools* dan bahasa pemrograman sehingga penting untuk dipelajari. Regex memungkinkan untuk melakukan pencarian pola yang memenuhi aturan tertentu dalam teks.

Terdapat beberapa aturan dalam *regular expression* yaitu sebagai berikut.

Character classes		**Groups & Lookaround**	
.	any character except newline	(abc)	capture group
\w \d \s	word, digit, whitespace	\1	backreference to group #1
\W \D \S	not word, digit, whitespace	(?:abc)	non-capturing group
[abc]	any of a, b, or c	(?=abc)	positive lookahead
[^abc]	not a, b, or c	(?!abc)	negative lookahead
[a-g]	character between a & g	**Quantifiers & Alternation**	
Anchors		a* a+ a?	0 or more, 1 or more, 0 or 1
^abc$	start / end of the string	a{5} a{2,}	exactly five, two or more
\b	word boundary	a{1,3}	between one & three
Escaped characters		a+? a{2,}?	match as few as possible
\. * \\	escaped special characters	ab|cd	match ab or cd
\t \n \r	tab, linefeed, carriage return		
\u00A9	unicode escaped ©		

Gambar 3. *Regex Cheat Sheet* (Sumber: [2])

Dengan menyusun sebuah string menggunakan regex yang tertera pada tabel di atas, kita dapat melakukan pencocokan terhadap pola yang memenuhi aturan tertentu, seperti melakukan verifikasi terhadap email yang valid, pola nomor telepon, string yang dapat ditulis menggunakan huruf kapital dan huruf kecil, ataupun string yang ditulis dengan sedikit kesalahan.

III. METODE

A. Program

Untuk mendukung pengumpulan data dan membantu analisis, akan dibuat sebuah program yang mengimplementasikan algoritma pattern matching dan regex yang dapat melakukan pattern matching terhadap dokumen karya ilmiah berformat PDF. Program ini merupakan program yang dikembangkan menggunakan bahasa pemrograman Python versi 3.11.9, yang dikenal dengan fleksibilitas dan kemampuannya dalam pengolahan teks serta dukungan pustaka yang kaya.

Tujuan utama dari program ini adalah melakukan perhitungan terhadap seluruh kata yang muncul pada sebuah dokumen karya ilmiah, kemudian mengurutkannya berdasarkan frekuensi kemunculannya. Proses ini melibatkan pemindaian teks dokumen secara menyeluruh untuk mengidentifikasi dan mencatat setiap kata yang digunakan menggunakan algoritma *pattern matching* dan regex, memungkinkan peneliti untuk memahami pola penggunaan kata dalam konteks akademik. Selain itu, program ini akan dilengkapi dengan fitur optimasi yang mengabaikan kata-kata umum seperti kata kerja dan kata yang sering muncul tetapi tidak relevan, seperti kata sambung atau preposisi. Fokus ini memungkinkan hasil pencarian menjadi lebih terarah kepada topik penelitian yang sedang diperiksa, meningkatkan kualitas dan relevansi analisis.

Selain fungsi dasar perhitungan kata, program ini juga dirancang untuk mencari kemunculan kata atau pola tertentu dalam dokumen. Program dapat mendeteksi dan mencatat posisi kemunculan setiap kata atau frasa yang diinginkan. Hal ini sangat berguna untuk analisis lebih mendalam, di mana peneliti dapat menilai seberapa sering dan di mana kata-kata penting muncul, serta mengevaluasi signifikansi kata-kata tersebut terhadap topik pembahasan dokumen. Dengan fitur ini, program dapat memberikan wawasan tentang bagaimana suatu topik dibahas dalam karya ilmiah, membantu peneliti memahami struktur dan fokus dari dokumen yang dianalisis. Teknik perhitungan kata tidak sekedar menghitung berapa kali jumlah kata muncul, melainkan menggabungkan kata-kata serupa yang merupakan *substring* dari kata yang lebih besar menjadi satu kesatuan, sehingga kata-kata yang mirip dihitung sebagai satu hal yang sama.

Dengan adanya program ini, diharapkan dapat mempermudah proses analisis dokumen karya ilmiah, menyediakan data yang lebih terstruktur, dan mendukung pengembangan penelitian lebih lanjut di bidang intelegensi buatan dan pengolahan teks. Program ini juga diharapkan menjadi alat yang berguna bagi akademisi untuk mengoptimalkan penelitian mereka, memungkinkan analisis yang lebih efisien dan mendalam terhadap literatur ilmiah.

B. Sampel Uji Coba

Untuk menguji kegunaan program, akan digunakan beberapa dokumen karya ilmiah mengenai konsep Strategi Algoritma yang terdaftar pada website Rinaldi Munir pada tahun 2023. Dokumen-dokumen ini dapat diakses melalui URL berikut: [Website Rinaldi Munir] (https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/Makalah2023.htm).

C. Langkah Uji Coba

Sebelum melakukan pengujian menggunakan dokumen karya ilmiah, penulis akan melakukan penilaian subjektif terhadap hal-hal penting yang dibahas dalam dokumen tersebut. Penilaian subjektif ini akan mencakup identifikasi topik utama, konsep kunci, dan terminologi yang sering digunakan dalam dokumen. Penulis akan membaca dan menganalisis konten secara manual untuk menentukan elemen-elemen yang dianggap paling relevan dan signifikan dalam konteks Strategi Algoritma.

Penilaian subjektif penulis ini akan digunakan sebagai perbandingan terhadap hasil yang diperoleh dari komputasi program. Dengan memiliki penilaian manusia sebagai acuan, kita dapat mengukur sejauh mana program mampu mendekati atau bahkan meniru cara berpikir manusia dalam mengidentifikasi dan menilai informasi penting dalam dokumen ilmiah. Perbandingan antara hasil penilaian subjektif dan hasil komputasi sangat penting karena algoritma ini didesain sebagai landasan yang nantinya akan dikembangkan menjadi intelegensi buatan yang menyerupai pola pikir manusia.

Dengan melakukan perbandingan ini, penulis dapat mengidentifikasi kekuatan dan kelemahan algoritma yang digunakan. Jika ada perbedaan signifikan antara penilaian subjektif dan hasil komputasi, analisis lebih lanjut dapat dilakukan untuk memahami penyebabnya dan melakukan penyesuaian pada algoritma. Tujuan akhir adalah menciptakan algoritma yang tidak hanya efisien dalam komputasi, tetapi juga memiliki tingkat akurasi tinggi dalam meniru pola pikir manusia dalam analisis dokumen ilmiah. Hal ini akan memberikan fondasi yang kuat untuk pengembangan lebih lanjut menuju sistem intelegensi buatan yang lebih canggih dan efektif dalam analisis dokumen karya ilmiah.

D. Prediksi Subjektif Penulis

Tabel 1. Prediksi Kata Kunci Penting Dalam Dokumen Karya Ilmiah

Judul Dokumen	Kata Kunci Penting
(1) Analisis Perbandingan Efisiensi Pencarian NIM Mahasiswa Teknik Informatika ITB dengan Metode Binary Search dan Interpolation Search	efisiensi, binary search, interpolation search.
(2) Pemecahan Sandi Caesar Dengan Bantuan Algoritma String Matching dan Brute-Force	sandi caesar, algoritma, string matching, brute force
(3) Perbandingan Algoritma Greedy dan Branch and Bound dalam Penjadwalan Job Freelance yang Memaksimalkan Keuntungan	algoritma, greedy, branch and bound, simpul, penjadwalan
(4) Penerapan Dynamic Programming dalam Menentukan Rute Chey-Yo Terpendek pada Robot ABU Robocon 2023	dynamic programming, penentuan rute, robot.
(5) Analisis Perbandingan Waktu Eksekusi de Bruijn Sequence Constructions dengan Greedy Algorithm dan Euler Cycle	waktu, eksekusi, de bruijn sequence constructions, greedy, algoritma, euler
(6) Application of Branch and Bound Algorithm to Solve Rider-Driver Batched Matching Problem	branch and bound, algorithm, rider-driver
(7) Pengembangan Algoritma Best First Search dalam Pencarian Solusi Game Tree	algoritma, best first search, simpul, solusi

Data yang disajikan pada tabel (1) merupakan prediksi subjektif penulis terhadap kata apa saja yang signifikan dibahas pada dokumen karya ilmiah terkait.

IV. Hasil dan Pembahasan

A. Hasil Uji Coba Program

Tabel 2. Hasil Uji Coba Program

Judul Dokumen	Keluaran Program
(1) Analisis Perbandingan Efisiensi Pencarian NIM Mahasiswa Teknik Informatika ITB dengan Metode Binary Search dan Interpolation Search	algoritma: 83 (Signifikansi: 2.95%) upalarik: 67 (Signifikansi: 2.38%) *binarysearchalgorithm: 64 (Signifikansi: 2.27%) diimplementasikan: 39 (Signifikansi: 1.39%) mahasiswa: 35 (Signifikansi: 1.24%) interpolasi: 35 (Signifikansi: 1.24%) *interpolationsearchalgorithm: 32 (Signifikansi: 1.14%) mengandung: 32 (Signifikansi: 1.14%) indeks: 31 (Signifikansi: 1.10%) elemen: 30 (Signifikansi: 1.07%)
(2) Pemecahan Sandi Caesar Dengan Bantuan Algoritma String Matching dan Brute-Force	menganalisis: 73 (Signifikansi: 2.15%) denga: 69 (Signifikansi: 2.03%) mengenkripsi: 58 (Signifikansi: 1.70%) -caesar: 57 (Signifikansi: 1.68%) abstract-sandi: 54 (Signifikansi: 1.59%) metodenya: 30 (Signifikansi: 0.88%) mencocokan: 30 (Signifikansi: 0.88%) mendekripsi: 30 (Signifikansi: 0.88%) algoritma: 29 (Signifikansi: 0.85%) menggeser: 26 (Signifikansi: 0.76%)
(3) Perbandingan Algoritma Greedy dan Branch and Bound dalam Penjadwalan Job Freelance yang Memaksimalkan Keuntungan	pengerjaan: 69 (Signifikansi: 2.05%) pekerjaannya: 57 (Signifikansi: 1.69%) bayaran: 52 (Signifikansi: 1.54%) algoritima: 52 (Signifikansi: 1.54%) *algoritma: 46 (Signifikansi: 1.37%) solusi: 41 (Signifikansi: 1.22%) simpul: 33 (Signifikansi: 0.98%) -greedy: 32 (Signifikansi: 0.95%) freelancer: 31 (Signifikansi: 0.92%) pesoalan: 31 (Signifikansi: 0.92%)
(4) Penerapan Dynamic Programming dalam Menentukan Rute Chey-Yo Terpendek pada Robot ABU Robocon 2023	programming: 36 (Signifikansi: 1.13%) denga: 36 (Signifikansi: 1.13%) dynamic: 30 (Signifikansi: 0.95%) tahapan: 28 (Signifikansi: 0.88%) permasalahan: 28 (Signifikansi: 0.88%) mempertimbangkan: 28 (Signifikansi: 0.88%) robotika: 26 (Signifikansi: 0.82%) makalah: 21 (Signifikansi: 0.66%) angkor: 21 (Signifikansi: 0.66%) optimalitas: 21 (Signifikansi: 0.66%)
(5) Analisis Perbandingan Waktu Eksekusi de Brujin Sequence Constructions dengan Greedy Algorithm dan Euler Cycle	konsekuensi: 60 (Signifikansi: 1.94%) sequencewithgreedy: 44 (Signifikansi: 1.42%) kombinasinya: 39 (Signifikansi: 1.26%) debruijnsequence: 31 (Signifikansi: 1.00%) rangkaian: 28 (Signifikansi: 0.90%) algoritma: 23 (Signifikansi: 0.74%) elemen: 23 (Signifikansi: 0.74%) simpul: 22 (Signifikansi: 0.71%) menggabungkan: 17 (Signifikansi: 0.55%) lakukan: 15 (Signifikansi: 0.48%)
(6) Application of Branch and Bound Algorithm to Solve Rider-Driver Batched Matching Problem	matched: 70 (Signifikansi: 2.22%) riders: 68 (Signifikansi: 2.15%) batchedmatchingbnb: 53 (Signifikansi: 1.68%) bounding: 50 (Signifikansi: 1.58%) drivers: 44 (Signifikansi: 1.39%) batching: 39 (Signifikansi: 1.23%) keywords-branch: 38 (Signifikansi: 1.20%) problem's: 34 (Signifikansi: 1.08%) algorithms-principles: 31 (Signifikansi: 0.98%) solutions: 24 (Signifikansi: 0.76%)
(7) Pengembangan Algoritma Best First Search dalam Pencarian Solusi Game Tree	simpul: 104 (Signifikansi: 4.09%) act-algoritma: 41 (Signifikansi: 1.61%) bernilai: 39 (Signifikansi: 1.53%) searching: 38 (Signifikansi: 1.49%) langkah: 29 (Signifikansi: 1.14%) merupaka: 26 (Signifikansi: 1.02%) solusi: 20 (Signifikansi: 0.79%) children: 20 (Signifikansi: 0.79%) number: 19 (Signifikansi: 0.75%) pemain: 19 (Signifikansi: 0.75%)

Data yang disajikan pada tabel (2) merupakan data keluaran program dari input file PDF yang bersesuaian. Keluaran diurutkan berdasarkan tingkat signifikansi dari tinggi ke rendah.

B. Waktu Eksekusi Algoritma

Tabel 3. Waktu Eksekusi Algoritma

Nomor Uji Coba	Waktu Eksekusi (ms)		Selisih (Keunggulan KMP)
	KMP	BM	
1	589	817	27.90%
2	1407	847	-66.11%
3	923	1209	23.65%
4	848	1129	24.88%
5	659	1185	44.38%
6	694	891	22.11%
7	520	922	43..60%
Rata-Rata	805.71	1000	17.20%

Data yang disajikan pada tabel (3) merupakan data waktu eksekusi program menggunakan kedua algoritma KMP dan BM terhadap input judul dokumen yang nomornya bersesuaian terhadap data yang tertera pada tabel (1) dan tabel (2).

Nilai selisih yang tertera merupakan hasil perhitungan (waktu BM - waktu KMP) / (waktu BM) * 100%.

C. Analisis Keluaran Program

Setelah menguji program dengan 7 masukan dokumen karya ilmiah yang berbeda, akan dilakukan analisis sederhana terhadap keluaran program dibandingkan dengan prediksi subjektif penulis. Analisis yang akan dilakukan adalah seberapa signifikan program menilai kemunculan sebuah kata pada dokumen ketimbang penilaian penulis yang merupakan seorang manusia.

Pada judul dokumen pertama yaitu *"Analisis Perbandingan Efisiensi Pencarian NIM Mahasiswa Teknik Informatika ITB dengan Metode Binary Search dan Interpolation Search"*, diprediksi kata yang signifikan antara lain 'efisiensi', '*binary search',* dan, '*interpolation search'*. Berdasarkan hasil keluaran program diperoleh kata 'algoritma' memiliki signifikansi tertinggi yaitu 2.95%, kata 'upalarik' sebesar 2.38%, kata '*binary search algorithm*' sebesar 2.27%, dan kata terkait 'interpolasi' sebesar 1.24+1.14%. Pada kasus ini hasil program sesuai harapan karena dapat menghasilkan keluaran yang diprediksi oleh manusia.

Pada judul dokumen kedua yaitu *"Pemecahan Sandi Caesar Dengan Bantuan Algoritma String Matching dan Brute-Force"*, diprediksi kata yang signifikan antara lain 'sandi caesar', 'algoritma', 'string matching', dan 'brute force'. Berdasarkan hasil keluaran program diperoleh kata 'menganalisis' dengan signifikansi sebesar 2.15%, kata 'enkripsi' sebesar 1.70%, kata 'caesar' sebesar 1.68%, 'abstract-sandi' sebesar 1.59%, dan kata 'algoritma' sebesar 0.85%. Pada kasus ini hasil program sesuai harapan karena dapat menghasilkan keluaran yang diprediksi oleh manusia.

Pada judul dokumen ketiga yaitu *"Perbandingan Algoritma Greedy dan Branch and Bound dalam Penjadwalan Job Freelance yang Memaksimalkan Keuntungan*", diprediksi kata yang signifikan antara lain 'algoritma', 'greedy', 'branch and bound', penjadwalan. Berdasarkan hasil keluaran program diperoleh kata 'pengerjaan' dengan signifikansi sebesar 2.05%, kata 'pekerjaannya' sebesar 1.69%, kata 'algoritma' sebesar 1.54+1.37%, kata 'simpul' sebesar 0.98%, dan 'greedy' sebesar 0.95%. Pada kasus ini hasil program kurang

memenuhi harapan karena dapat menghasilkan keluaran yang mirip dengan perlakuan manusia secara utuh.

Pada judul dokumen keempat yaitu *"Penerapan Dynamic Programming dalam Menentukan Rute Chey-Yo Terpendek pada Robot ABU Robocon 2023"*, diprediksi kata yang signifikan antara lain 'dynamic programming', 'penentuan rute', dan 'robot'. Berdasarkan hasil keluaran program diperoleh kata 'programming' dengan signifikansi sebesar 1.13%, kata 'dynamic' sebesar 0.95%, dan kata 'robotika' sebesar 0.82%. Pada kasus ini hasil program sesuai harapan karena dapat menghasilkan keluaran yang diprediksi oleh manusia.

Pada judul dokumen kelima yaitu *"Analisis Perbandingan Waktu Eksekusi de Brujin Sequence Constructions dengan Greedy Algorithm dan Euler Cycle"*, diprediksi kata yang signifikan antara lain 'waktu, eksekusi, de bruijn sequence constructions', 'algoritma', dan 'euler'. Berdasarkan hasil keluaran program diperoleh kata 'konsekuensi' yang memiliki signifikansi sebesar 1.94%, kata 'sequencewithgreedy' sebesar 1.42%, kata 'kombinasi' sebear 1.26%, kata 'debruijn sequence' sebesar 1%, dan kata 'algoritma' sebesar 0.74%. Pada kasus ini hasil program sesuai harapan karena dapat menghasilkan keluaran yang diprediksi oleh manusia.

Pada judul dokumen keenam yaitu *"Application of Branch and Bound Algorithm to Solve Rider-Driver Batched Matching Problem"*, diprediksi kata yang signifikan antara lain 'branch and bound', 'algorithm', 'rider-driver'. Berdasarkan hasil keluaran program diperoleh kata 'matched' dengan signifikansi 2.22%, kata 'riders' sebesar 2.15%, kata 'batchedmatchingbnb' sebesar 1.68%, kata 'bounding' sebesar 1.58%, dan kata 'drivers' sebesar 1.39%. Pada kasus ini hasil program sesuai harapan karena dapat menghasilkan keluaran yang diprediksi oleh manusia.

Pada judul dokumen ketujuh yaitu *"Pengembangan Algoritma Best First Search dalam Pencarian Solusi Game Tree"*, diprediksi kata yang signifikan antara lain 'algoritma', 'best first search', 'simpul', dan 'solusi'. Berdasarkan hasil keluaran program diperoleh kata 'simpul' dengan signifikansi sebesar 4.09%, kata 'algoritma' sebesar 1.61%, dan kata 'solusi' sebesar 0.79%. Pada kasus ini hasil program sesuai harapan karena dapat menghasilkan keluaran yang diprediksi oleh manusia.

Berdasarkan ketujuh hasil analisis, dapat dikatakan bahwa hasil dari algoritma memiliki hasil yang cukup memuaskan karena dapat memperoleh enam hasil yang serupa dengan prediksi manusia dari tujuh kali percobaan. Nilai kesuksesan dari algoritma ini berdasarkan uji coba yang dilakukan adalah 6/7 *x* 100% atau 85.714%.

D. Analisis Waktu Eksekusi

Untuk membandingkan keunggulan antara algoritma pattern matching milik Knuth-Morris-Pratt dengan milik Boyer-Moore, eksperimen ini dilakukan menggunakan kedua algoritma tersebut untuk dibandingkan waktu eksekusinya. Analisis waktu eksekusi dilakukan dengan menjalankan kedua algoritma pada berbagai kasus uji coba. Hasil dari analisis waktu eksekusi kemudian tertera pada tabel (3).

Berdasarkan hasil yang tertera, algoritma KMP unggul pada 6 dari 7 kasus uji coba. Analisis menunjukkan bahwa algoritma KMP memiliki keunggulan rata-rata sebesar 17.20%. Maka dari itu, dapat dikatakan bahwa pada kasus menganalisis teks yang cukup besar, dalam kasus ini dokumen karya ilmiah, algoritma KMP lebih diunggulkan sesuai dengan teori kompleksitas waktunya yang linear..

V. Kesimpulan

Berdasarkan analisis yang telah dilakukan terhadap hasil eksperimen. Hasil perhitungan signifikansi yang diperoleh berdasarkan komputasi menunjukan hasil yang menyerupai prediksi manusia. Maka dari itu, hipotesis tentang menggunakan algoritma *pattern matching* khususnya KMP dan BM dengan bantuan regex untuk menganalisis kemunculan informasi penting pada dokumen karya ilmiah untuk dijadikan landasan *machine learning* intelegensi buatan tidaklah jauh dari kenyataan.

Selain itu, disimpulkan pula bahwa dalam menganalisis teks yang cukup panjang seperti dokumen karya ilmiah, algoritma milik KMP diunggulkan ketimbang BM, hasil yang sesuai dengan teori kompleksitas waktunya yang linear.

VI. Saran

Saran untuk peneliti selanjutnya adalah menganalisis teks pada dokumen karya ilmiah menggunakan teknik yang lebih maju dengan pendekatan *machine learning* secara langsung, sehingga dapat menyimpulkan topik pembahasan pada suatu paragraf di dalam dokumen karya ilmiah tersebut.

VII. Ucapan Terimakasih

Terima kasih kepada Tuhan Yang Maha Esa karena dengan rahmat dan hidayahnya makalah berjudul "Pemanfaatan Algoritma Pattern Matching dan Regex Dalam Pencarian Informasi Pada Dokumen Karya Ilmiah" dapat terselesaikan tepat waktu. Selanjutnya saya juga berterima kasih kepada Bpk Ir. Rila Mandala, M.Eng., Ph.D. dan Bpk Monterico Adrian, S.T, M.T yang telah membimbing mata kuliah IF2211 Strategi Algoritma selama satu semester ini serta Bpk Bpk Dr. Ir. Rinaldi Munir, S.T, M.T selaku dosen koordinator mata kuliah ini.

Pranala Video Youtube

https://www.youtube.com/watch?v=tBL8grLsyC8&ab_channel=chomusuke

Pranala Repository GitHub

https://github.com/rafimaliki/13522137-Makalah-Stima-2024

Daftar Pustaka

[1] Rinaldi Munir. 2021. *"Pencocokan String (String/Pattern Matching)"* . Diakses pada 12 Juni 2024 https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

[2] Rinaldi Munir. 2019. *"String Matching dengan Regular Expression"* . Diakses pada 12 Juni 2024 dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/String-Matching-dengan-Regex-2019.pdf

[3] Dan's Tools. *"RegEx Pal"*. Diakses pada 12 Juni 2024 dari https://www.regexpal.com/.

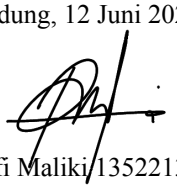
[4] Nature.Com. *"Scientific Papers"* Diakses pada 12 Juni 2024 dari https://www.nature.com/scitable/topicpage/scientific-papers-13815490/#:~:text=Scientific%20papers%20are%20for%20sharing,builds%20upon%20that%20of%20others.

[5] HyperSkill. *"Knuth-Morris-Pratt algoritm"*. Diakses pada 12 Juni 2024 dari https://hyperskill.org/learn/step/34945.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Juni 2024



Ahmad Rafi Maliki/13522137