APLIKASI ANALISIS LOG SISTEM UNTUK DETEKSI POLA ERROR MENGGUNAKAN ALGORITMA STRING MATCHING KNUTH-MORRIS-PRATT (KMP)

Oleh :

(1) Aprilianingsih (105841100323)
(2) Nur Fahira Nurdin (105841100723)

Abstrak

Log sistem merupakan sumber informasi vital yang mencatat seluruh aktivitas komputer. Namun, pertumbuhan volume data log yang masif pada infrastruktur modern membuat analisis manual menjadi tidak efisien dan rentan terhadap kesalahan manusia, Pendekatan untuk mengotomatisasi analisis log sangat diperlukan untuk mengatasi besarnya jumlah data log dan mengurangi beban kerja operator. Penelitian ini bertujuan merancang aplikasi analisis log untuk mendeteksi pola *error* menggunakan algoritma *string matching* Knuth-Morris-Pratt (KMP). Algoritma ini mengurangi waktu perbandingan dengan menghilangkan penelusuran balik backtracking pada pointer teks, menjadikannya efisien untuk sistem deteksi intrusi. Metode penelitian dilakukan secara eksperimental dengan mengembangkan aplikasi berbasis Python yang memindai file log untuk kata kunci kritis seperti "error", "failed", dan "warning". Hasil pengujian menunjukkan aplikasi mampu mendeteksi anomali dengan akurat dan cepat. Deteksi tepat waktu terhadap perilaku abnormal sangat penting untuk mencegah dampak negatif pada layanan, sehingga membantu membangun sistem yang aman dan terpercaya.

Kata Kunci: log sistem, deteksi error, string matching, analisis log, Python

Abstract

System logs are a vital source of information that records all computer activity. However, the massive growth in log data volumes in modern infrastructure makes manual analysis inefficient and prone to human error. Approaches to automate log analysis are urgently needed to address the large volume of log data and reduce operator workload. This study aims to design a log analysis application to detect error patterns using the Knuth-Morris-Pratt (KMP) string matching algorithm. This algorithm reduces comparison time by eliminating backtracking on text pointers, making it efficient for intrusion detection systems. The research method was carried out experimentally by developing a Python-based application that scans log files for critical keywords such as "error," "failed," and "exception." Test results showed the application was able to detect anomalies accurately and quickly. Timely detection of abnormal behavior is crucial to prevent negative impacts on services, thus helping to build secure and reliable systems.

Keyword: system log, error detection, string matching, log analysis, Python

PENDAHULUAN

Perkembangan teknologi informasi saat ini menyebabkan lonjakan volume data log sistem secara eksponensial. Padahal, log memegang peranan vital sebagai satu-satunya rekaman data yang menyediakan wawasan rinci mengenai perilaku *runtime* perangkat lunak guna menjamin keandalan system (He et al., 2022). Namun, karakteristik data log modern yang tidak terstruktur dan bervolume besar membuat metode inspeksi manual menjadi tidak efisien, mustahil dilakukan secara *real-time*, dan sangat rentan terhadap *human error* (Landauer et al., 2023).

Untuk mengatasi tantangan tersebut, pendekatan berbasis *Deep Learning* seperti *Long Short-Term Memory* (LSTM) memang terbukti unggul dalam menangkap anomali pada urutan log yang dinamis (Himler et al., 2024). Akan tetapi, metode tersebut relatif kompleks karena menuntut

sumber daya komputasi yang tinggi dan seringkali terlalu berlebihan (*overkill*) dibandingkan metode standar yang menyeimbangkan kecepatan dan efisiensi memori(Aksa et al., 2021). Oleh karena itu, diperlukan metode yang lebih sederhana namun efektif. Salah satu pendekatan yang paling relevan untuk pencarian pola *error* yang bersifat pasti adalah *exact string matching* (R et al., 2021). Dalam konteks pencarian data pada aplikasi pengarsipan, KMP digunakan sebagai metode pencarian string eksak yang sistematis. Meskipun terdapat berbagai metode pencarian lain, KMP tetap menjadi algoritma yang andal untuk menemukan data yang presisi dengan waktu proses yang relatif cepat dan stabil dalam menangani ketidakcocokan karakter (*NurulHuda*, n.d.).

Penelitian ini menerapkan algoritma Knuth-Morris-Pratt (KMP) sebagai solusi efisiensi pencarian pola. Berbeda dengan metode konvensional, KMP menawarkan kecepatan tinggi karena kemampuannya menghilangkan penelusuran balik (*backtracking*) pada pointer teks menggunakan aturan lompatan heuristik berbasis prefix (Zhang, 2022). Efisiensi ini sangat bermanfaat dalam konteks keamanan siber, khususnya untuk mendeteksi pola perilaku mencurigakan pada layanan SSH (*Secure Shell*) dan menghindari perbandingan ulang karakter (Sano & Rasyid, n.d.). Validitas efisiensi KMP ini juga telah terbukti secara empiris mampu meminimalkan waktu respons pencarian dibandingkan algoritma konvensional (Marbun et al., 2024).

Tujuan dari penelitian ini adalah membangun aplikasi analisis log sistem untuk mendeteksi pola *error* menggunakan metode *string matching* KMP berbasis Python. Pendekatan otomatisasi ini sangat diperlukan tidak hanya untuk menangani besarnya data, tetapi juga untuk mengurangi beban kerja kognitif operator secara signifikan (Landauer et al., 2023). Aplikasi ini diharapkan dapat membantu administrator sistem dalam mengidentifikasi *error* dengan cepat serta meningkatkan efisiensi proses *troubleshooting*. Lebih jauh lagi, deteksi dini terhadap perilaku abnormal ini merupakan langkah krusial dalam mencegah dampak negatif layanan guna membangun sistem yang aman dan terpercaya. Lebih lanjut, perbandingan string memainkan peran sentral dalam forensik digital, di mana mendeteksi manipulasi struktural dalam data berbasis teks atau log sangat penting untuk mengidentifikasi strategi pengaburan informasi oleh penyerang (Cantone et al., 2025).

METODE PENELITIAN

Penelitian ini menggunakan metode eksperimental dengan pendekatan pengembangan sistem (*System Development Life Cycle*). Metode ini bertujuan untuk menguji efektivitas dan efisiensi algoritma *string matching* dalam mendeteksi pola anomali pada log system dengan waktu respons yang cepat (high-speed response). Pendekatan eksperimental dipilih untuk mengukur performa komputasi algoritma ketika dihadapkan pada variasi volume data log yang berbeda.

1. **Jenis Penelitian**

   Jenis penelitian ini dikategorikan sebagai penelitian terapan (*applied research*). Penelitian berfokus pada penerapan algoritma teoretis Knuth-Morris-Pratt (KMP) untuk memecahkan masalah praktis dalam infrastruktur TI, yaitu inefisiensi dan keterlambatan deteksi kesalahan pada log server. Penelitian ini juga bersifat kuantitatif karena melibatkan pengukuran waktu eksekusi (*runtime*) dan akurasi deteksi pola.

2. **Alat dan Bahan Penelitian**

   Untuk mendukung implementasi dan pengujian sistem, penelitian ini menggunakan perangkat keras dan perangkat lunak dengan spesifikasi sebagai berikut:

   a. Perangkat Keras: Laptop dengan prosesor AMD Ryzen 5 7535HS with Radeon Graphics (3.30 GHz), RAM 8GB. Sistem operasi yang digunakan adalah Windows 11 64-bit.
   b. Perangkat Lunak:
      1) Bahasa Pemrograman: Python 3.10 digunakan sebagai fondasi utama. Pemilihan Python didasarkan pada keandalannya dalam pengembangan alat monitoring keamanan sistem dan manipulasi data teks yang efisien, sebagaimana diterapkan dalam pengembangan sistem IPS (*Intrusion Prevention System*) (Hardjianto, 2022).

2) Framework Antarmuka: Streamlit digunakan untuk membangun antarmuka pengguna (User Interface) berbasis web yang interaktif, memungkinkan pengguna mengunggah file .log secara langsung.
3) Pustaka Pendukung : Pandas digunakan untuk manajemen struktur data hasil deteksi dalam bentuk tabel (*DataFrame*).

3. **Objek Penelitian**

Objek penelitian ini adalah data log sistem server (*server logs*) yang bersifat tidak terstruktur atau semi-terstruktur. Sesuai dengan karakteristik yang dijelaskan oleh(Landauer et al., 2023), data log ini mengandung ribuan baris teks dengan ketergantungan semantik yang rumit, mencakup informasi *timestamp*, *IP address*, dan pesan status. Format log yang bervariasi ini menuntut penggunaan algoritma pencocokan pola yang presisi, karena metode manual rentan terhadap kesalahan interpretasi terutama pada volume data yang besar(He et al., 2022)

4. **Algoritma yang Digunakan**

Algoritma KMP diimplementasikan untuk melakukan pencarian string eksak (exact matching). Algoritma ini dipilih karena kompleksitas waktu O(n+m) yang linear. Hal ini memastikan perilaku waktu linier di seluruh kasus terbaik, rata-rata, dan terburuk, menjadikan KMP sangat dapat diprediksi dan andal dibandingkan algoritma lain yang mungkin mengalami degradasi performa pada input tertentu (Gor & Bhensdadia, 2025). Mekanisme utama yang diterapkan adalah penggunaan tabel LPS (Longest Prefix Suffix) yang dihitung pada tahap preprocessing. Tabel ini memungkinkan sistem menentukan 'lompatan' karakter ketika terjadi ketidakcocokan, sehingga mencegah terjadinya penelusuran balik (backtracking) yang memboroskan sumber daya komputasi (Angelina et al., 2023).Tabel ini memungkinkan algoritma untuk melewati perbandingan karakter yang tidak perlu, sehingga sangat efisien dalam memproses teks dalam jumlah besar tanpa harus melakukan penelusuran balik (Tan, 2025).
Efektivitas implementasi KMP juga terbukti mampu meningkatkan kinerja sistem informasi dalam menangani pencarian data. Berdasarkan pengujian kinerja pada sistem pencarian data buku, algoritma ini menunjukkan stabilitas waktu eksekusi yang sangat cepat. Hasil ini menegaskan bahwa algoritma KMP mampu bekerja secara presisi dan efisien dalam menemukan pola kata kunci yang relevan tanpa membebani kinerja sistem secara keseluruhan (Kurniawan & Indrianti, 2022).

5. **Teknik Pengumpulan Data**
Pengumpulan data dilakukan dengan metode simulasi dan observasi.

a. Sumber Data: Sampel data log diambil dari repositori log server Apache/Nginx dan simulasi system event yang mengandung campuran aktivitas normal dan abnormal.
b. Definisi Pola: Penentuan kata kunci (keywords) yang menjadi target pencarian didasarkan pada standar protokol HTTP dan pesan sistem, yaitu: "ERROR", "FAILED", dan "WARNING".

6. **Langkah Penelitian**
Tahapan penelitian dirancang secara sistematis mengikuti alur pemrosesan data pada aplikasi yang dibangun:

a. Input Data (File Upload): Sistem menerima input file log dari pengguna melalui antarmuka Streamlit. File dibaca dan dikonversi (*decoding*) menjadi format string UTF-8 untuk memastikan seluruh karakter terbaca dengan benar.
b. Preprocessing & LPS Computation: Sebelum pencarian dimulai, sistem menghitung tabel LPS (*Failure Function*) berdasarkan kata kunci yang dimasukkan (misal: "ERROR

dan WARNING"). Langkah ini krusial untuk memetakan pola pengulangan dalam kata kunci guna mempercepat proses pencocokan.

Karakter	Substring	LPS Value	Penjelasan
E	E	0	Tidak ada prefiks/sufiks (karakter tunggal).
R	ER	0	Prefiks "E", Sufiks "R". Tidak sama.
R	ERR	0	Prefiks "E", Sufiks "R". Tidak sama.
O	ERRO	0	Prefiks "E", Sufiks "O". Tidak sama.
R	ERROR	0	Prefiks "E", Sufiks "R". Tidak sama.

Tabel 1.1 Perhitungan LPS untuk Kata Kunci "ERROR"

Karakter	Substring	LPS Value	Penjelasan
W	W	0	Tidak ada prefiks/sufiks (karakter tunggal).
A	WA	0	Prefiks "W", Sufiks "A". Tidak sama.
R	WAR	0	Prefiks "E", Sufiks "R". Tidak sama.
N	WARN	0	Prefiks "E", Sufiks "N". Tidak sama.
I	WARNI	0	Prefiks "E", Sufiks "I". Tidak sama.
N	WARNIN	0	Prefiks "E", Sufiks "N". Tidak sama.
G	WARNING	0	Prefiks "E", Sufiks "G". Tidak sama.

Tabel 1.2 Perhitungan LPS untuk Kata Kunci "WARNING"

c. Proses Pencocokan String (Matching Process): Algoritma KMP memindai setiap baris log. Jika pola ditemukan, sistem akan mencatat nomor baris dan isi pesan log tersebut ke dalam memori sementara (*list*). Waktu mulai dan selesai proses ini dicatat untuk menghitung durasi eksekusi.

Langkah	Posisi Text (*i*)	Karakter Text	Posisi Pola (*j*)	Karakter Pola	Status	Aksi Pergeseran
1	0	E	0	E	Cocok	Lanjut ke *i*+1, *j*+1
2	1	R	1	R	Cocok	Lanjut ke *i*+1, *j*+1
3	2	R	2	R	Cocok	Lanjut ke *i*+1, *j*+1
4	3	(spasi)	3	O	Tidak Cocok	Lihat LPS index sebelumnya (*j*=2). LPS[2]=0. Reset *j* ke 0.

5	3	(spasi)	0	E	Tidak Cocok	Text maju (*i*+1), *j* tetap 0.
6	4	E	0	E	Cocok	Lanjut ke *i*+1, *j*+1
7	5	R	1	R	Cocok	Lanjut ke *i*+1, *j*+1
8	6	R	2	R	Cocok	Lanjut ke *i*+1, *j*+1
9	7	O	3	O	Cocok	Lanjut ke *i*+1, *j*+1
10	8	R	4	R	Cocok	POLA DITEMUKAN

Tabel 1.3 Simulasi Pergeseran Algoritma KMP pada Teks "ERR ERROR"

d. Visualisasi & Pelaporan: Hasil deteksi dikonversi menjadi tabel data (*DataFrame*) dan ditampilkan ke layar dashboard. Sistem juga menyajikan ringkasan statistik berupa total baris yang dipindai, jumlah *error* yang ditemukan, dan waktu proses dalam satuan detik, yang berguna untuk evaluasi kinerja sistem.

HASIL DAN PEMBAHASAN

1. Analisis Kinerja Deteksi Hasil pengujian menunjukkan bahwa aplikasi analisis log sistem yang dikembangkan mampu mendeteksi pola *error* secara akurat menggunakan algoritma Knuth-Morris-Pratt (KMP). Sebagaimana ditampilkan pada Gambar 1.1, sistem berhasil mengidentifikasi seluruh baris log yang mengandung kata kunci "ERROR" tanpa kesalahan (*false negative*).Secara performa komputasi, aplikasi menunjukkan efisiensi waktu yang signifikan. Berdasarkan skenario pengujian pada file log dengan berbagai jumlah total baris data.

NO	Jumlah Baris	Keyword	Waktu Proses (detik)
1	122	Error	0.0239 detik
2	122	Warning	0.0238 detik
3	122	Failed	0.0297 detik
4	244	Error	0.0297 detik
5	244	Warning	0.0352 detik

6	244	Failed	0.0321 detik
7	488	Error	0.0346 detik
8	488	Warning	0.0397 detik
9	488	Failed	0.0394 detik
10	976	Error	0.0427 detik

Tabel 1.4 Hasil Pengujian Waktu Proses Deteksi

Pada table diatas menyajikan hasil pengukuran waktu eksekusi algoritma terhadap variasi volume data log. Hasil pengujian menunjukkan bahwa peningkatan jumlah baris data (dari 122 menjadi 244 baris) hanya menyebabkan kenaikan waktu proses yang relatif kecil, yakni pada kisaran 0,01 detik. Fenomena ini mengonfirmasi stabilitas algoritma KMP yang memiliki kompleksitas waktu linear, di mana performa sistem tetap terjaga cepat dan tidak membebani memori meskipun beban data yang diperiksa bertambah dua kali lipat.

2. Validasi dengan Penelitian Terdahulu Hasil efisiensi ini sejalan dengan temuan empiris pada domain sistem informasi lainnya. Sebagaimana dibuktikan oleh (Marbun et al., 2024) pada sistem *e-katalog* perpustakaan, implementasi KMP terbukti signifikan meminimalkan waktu respons pencarian dibandingkan metode konvensional. Algoritma ini dinilai efisien karena kemampuannya meminimalkan jumlah perbandingan pola terhadap string input, yang mempercepat proses pencarian dibandingkan metode konvensional(Permatasari et al., 2024). Konsistensi performa ini juga dikonfirmasi oleh (Diti et al., 2022) yang menemukan bahwa penerapan algoritma Knuth-Morris-Pratt dalam sistem informasi terbukti mampu membuat proses penelusuran data menjadi lebih responsif dibandingkan pencarian manual.

3. Perbandingan dengan Pendekatan *Deep Learning*, Dalam konteks deteksi anomali modern, penelitian ini menawarkan perspektif efisiensi sumber daya. Meskipun pendekatan berbasis *Deep Learning* seperti model LSTM (*Long Short-Term Memory*) menawarkan kemampuan untuk menangkap anomali pada urutan log yang sangat dinamis dan kompleks (Himler et al., 2024), metode tersebut menuntut biaya komputasi yang tinggi dan proses pelatihan model yang rumit. Bahkan, tantangan privasi pada metode *Deep Learning* seringkali memaksa penggunaan arsitektur tambahan seperti *Federated Learning* yang menambah kompleksitas sistem (Himler et al., 2024).

4. Implikasi Praktis Aplikasi yang dihasilkan tidak hanya berfungsi sebagai alat pencarian, tetapi juga mendukung aspek keamanan. Kemampuan untuk mengidentifikasi perilaku abnormal dengan waktu respons instan (rapid response) adalah langkah fundamental dalam memitigasi risiko keamanan siber (Hardjianto, 2022). Dengan antarmuka berbasis Streamlit yang interaktif, solusi ini menjembatani kesenjangan antara kompleksitas data log mentah dengan kebutuhan administrator akan informasi yang dapat segera ditindaklanjuti.

Gambar 1.1 Pencarian dengan kata kunci (Error)

Pada gambar ini memperlihatkan antarmuka sistem setelah proses analisis selesai dengan kata kunci 'ERROR'. Sistem menampilkan notifikasi sukses yang menunjukkan jumlah baris yang dipindai serta total anomali yang ditemukan (29 masalah). Di bagian ini, pengguna dapat melihat secara langsung ringkasan kinerja deteksi.

Gambar 1.2 Pencarian dengan kata kunci (Warning)

Pada pengujian kedua pada gambar diatas, sistem diuji menggunakan kata kunci 'WARNING'. Hasil menunjukkan bahwa algoritma mampu beradaptasi dengan pola input yang berbeda dan tetap memberikan hasil deteksi yang presisi, menangkap peringatan dini seperti penggunaan CPU tinggi atau disk yang hampir penuh.

	Baris Ke-	Detail Pesan Log
0	6	2024-01-01 08:05:25 ERROR Database connection failed
1	8	2024-01-01 08:07:35 ERROR Timeout while connecting to database
2	15	2024-01-01 08:13:02 ERROR File configuration not found
3	20	2024-01-01 08:18:50 ERROR Failed to write log file
4	25	2024-01-01 08:23:15 ERROR Network connection lost
5	29	2024-01-01 08:27:35 ERROR Unauthorized access attempt
6	35	2024-01-01 08:33:00 ERROR API response timeout
7	39	2024-01-01 08:37:20 ERROR Failed to allocate memory
8	43	2024-01-01 08:41:40 ERROR System resource limit exceeded
9	51	2024-01-01 08:49:15 ERROR Overheating detected on CPU

Gambar 1.3 Hasil Tabel Pencarian dengan kata kunci (Error)

Gambar diatas menyajikan detail hasil pencarian dalam format tabel DataFrame. Setiap baris log yang mengandung kata 'ERROR' diekstraksi lengkap dengan nomor baris aslinya. Informasi detail seperti 'Database connection failed' atau 'Timeout' ditampilkan dengan jelas untuk memudahkan administrator melakukan troubleshooting pada baris yang spesifik.

	Baris Ke-	Detail Pesan Log
0	4	2024-01-01 08:03:12 WARNING Penggunaan CPU mencapai 75%
1	9	2024-01-01 08:08:40 WARNING Penggunaan memori mencapai 80%
2	17	2024-01-01 08:15:20 WARNING Disk hampir penuh
3	24	2024-01-01 08:22:10 WARNING Koneksi jaringan tidak stabil
4	30	2024-01-01 08:28:40 WARNING Percobaan login gagal 3 kali
5	37	2024-01-01 08:35:10 WARNING Latency jaringan meningkat
6	42	2024-01-01 08:40:35 WARNING Penggunaan memori mencapai 85%
7	50	2024-01-01 08:48:10 WARNING Suhu server meningkat
8	59	2024-01-01 08:03:12 WARNING Penggunaan CPU mencapai 75%
9	64	2024-01-01 08:08:40 WARNING Penggunaan memori mencapai 80%

Gambar 1.4 Hasil Tabel Pencarian dengan kata kunci (Warning)

Serupa dengan deteksi error, Gambar 1.4 menampilkan rincian log untuk kategori 'WARNING'. Tampilan ini menegaskan kemampuan aplikasi dalam mengelompokkan jenis anomali, memungkinkan operator untuk memprioritaskan penanganan masalah berdasarkan tingkat keparahannya (misalnya membedakan antara error kritis dengan sekadar warning kapasitas memori).

SIMPULAN

Berdasarkan hasil penelitian dan pengujian sistem yang telah dilakukan, dapat disimpulkan bahwa aplikasi analisis log sistem yang dibangun menggunakan algoritma *String Matching* Knuth-Morris-Pratt (KMP) terbukti sangat efektif. Secara empiris, aplikasi mampu mendeteksi pola *error* dengan rata-rata waktu eksekusi yang sangat cepat (di bawah 0.5 detik) untuk file log berukuran sedang. Implementasi KMP berhasil memvalidasi teori efisiensi O(n+m) dengan menghilangkan kelemahan penelusuran balik (*backtracking*) yang ada pada metode naif, sehingga performa sistem tetap stabil dan linear meskipun dihadapkan pada lonjakan volume data.

Dari sisi implikasi praktis, integrasi bahasa pemrograman Python dan kerangka kerja Streamlit berhasil menciptakan alat pemantauan yang tidak hanya ringan (*lightweight*), tetapi juga krusial untuk aspek keamanan siber. Kemampuan sistem dalam memberikan deteksi dini secara *real-time* menjawab tantangan fatalitas *human error* dalam inspeksi manual serta memitigasi risiko serangan pada port server (seperti SSH). Hal ini secara langsung berkontribusi pada pengurangan beban kerja kognitif administrator dan peningkatan ketersediaan layanan (*availability*) infrastruktur TI.

Untuk pengembangan penelitian selanjutnya, disarankan agar sistem ini tidak hanya bergantung pada pencarian kata kunci statis, tetapi mulai diintegrasikan dengan metode *Deep Learning* seperti LSTM (*Long Short-Term Memory*) untuk mendeteksi anomali perilaku yang tidak terdefinisi (*zero-day anomalies*). Selain itu, eksplorasi teknik pemrosesan paralel menggunakan kerangka kerja *Big Data* (seperti Hadoop MapReduce) sangat direkomendasikan untuk meningkatkan skalabilitas sistem dalam menangani arsip log historis berskala masif.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada pihak-pihak yang telah membantu dalam penyusunan dan pelaksanaan penelitian ini.

DAFTAR PUSTAKA

Aksa, M., Rashid, J., Nisar, M. W., Mahmood, T., Kwon, H. Y., & Hussain, A. (2021). Bitmapaligner: Bit-parallelism string matching withmapreduce and hadoop. *Computers, Materials and Continua*, *68*(3), 3931–3946. https://doi.org/10.32604/cmc.2021.016081

Angelina, R., Hutabarat, P., Hutapea, J. S., Marlina, D., & Lubis, M. (2023). PENERAPAN ALGORITMA STRING MATCHING DALAM PENCOCOKAN DATA STRING. In *Jurnal Teknik Informatika* (Vol. 15, Issue 2).

Cantone, D., Faro, S., & Pavone, A. (2025). Approximate String Matching with Non-Overlapping Adjacent Unbalanced Translocations. *Mathematics*, *13*(13). https://doi.org/10.3390/math13132103

Diti, N. A., Khasanah, F. N., & Aida Fitriyani. (2022). Penerapan Algoritma Pencarian Knuth-Morris-Pratt dalam Sistem Informasi Perpustakaan pada SMAN 1 Babelan. *Journal of Students' Research in Computer Science*, *3*(2), 171–184. https://doi.org/10.31599/jsrcs.v3i2.1507

Gor, A., & Bhensdadia, C. K. (2025). Comparative Analysis of Classical String Matching Algorithms with Insights into Applications, Parallel Processing, and Big Data Frameworks. In *International Journal of Computer Applications* (Vol. 187, Issue 53).

Hardjianto, M. (2022). Sistem Monitoring Serangan Ssh Dengan Metode Intrusion Prevention System (IPS) Fail2ban Menggunakan Python Pada Sistem Operasi Linux. *Jurnal TICOM: Technology of Information and Communication*, *11*(1).

He, S., He, P., Chen, Z., Yang, T., Su, Y., & Lyu, M. R. (2022). A Survey on Automated Log Analysis for Reliability Engineering. In *ACM Computing Surveys* (Vol. 54, Issue 6). Association for Computing Machinery. https://doi.org/10.1145/3460345

Himler, P., Landauer, M., Skopik, F., & Wurzenberger, M. (2024). Anomaly detection in log-event sequences: A federated deep learning approach and open challenges. *Machine Learning with Applications*, *16*, 100554. https://doi.org/10.1016/j.mlwa.2024.100554

Kurniawan, H., & Indrianti, V. (2022). KLIK: Kajian Ilmiah Informatika dan Komputer Implementasi Algoritma Knuth Morris Pratt untuk Pencariaan Data Buku Pada Sistem Informasi Perpustakaan. *Media Online*, *3*(3), 291–297. https://djournals.com/klik

Landauer, M., Onder, S., Skopik, F., & Wurzenberger, M. (2023). Deep learning for anomaly detection in log data: A survey. *Machine Learning with Applications*, *12*, 100470. https://doi.org/10.1016/j.mlwa.2023.100470

Marbun, N., Rozy, A., Pasaribu, S. A., Bu'ulolo, E., Purba, B., Hasibuan, N. N., Riansyah, M., & Abstrak, I. A. (2024). *KETIK : Jurnal Informatika Implementasi Algoritma Knuth-Morris-Pratt Pada E-Katalog Perpustakaan*. https://doi.org/10.70404/ketik.v2i02.143

NurulHuda. (n.d.).

Permatasari, H., Purwanto, E., & Triyono, T. (2024). Prototipe Pencarian Berkas Kinerja Menggunakan Algoritma Knuth Morris Pratt (Studi Kasus pada Lembaga Amil Zakat). *Jurnal Teknologi Sistem Informasi Dan Aplikasi*, *7*(1), 109–115. https://doi.org/10.32493/jtsi.v7i1.34500

R, C. A., Sunil Kumar, G., & Swamy, M. B. (2021). Survey and Comparison of String Matching Algorithms. In *Turkish Journal of Computer and Mathematics Education* (Vol. 12, Issue 12).

Tan, J. (2025). *Performance Comparison of Different Code Implementations of the KMP Algorithm*. https://doi.org/10.54254/2755-2721/115/2025.18467

Zhang, Z. (2022). Review on String-Matching Algorithm. *SHS Web of Conferences*, *144*, 03018. https://doi.org/10.1051/shsconf/202214403018