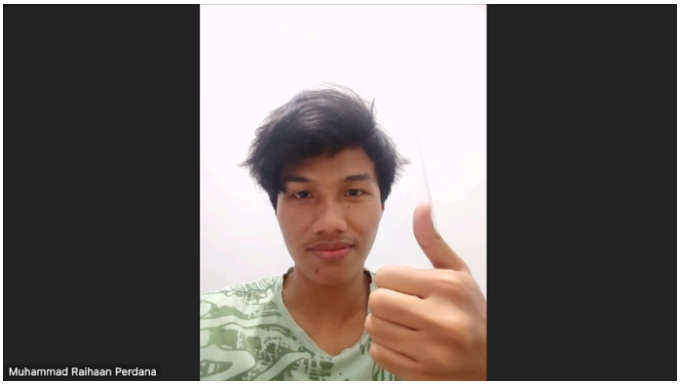
Tugas Besar 3 IF2211 Strategi Algoritma

Pemanfaatan Pattern Matching untuk Membangun Sistem ATS (Applicant Tracking System) Berbasis CV Digital



Oleh:

Kelompok 3 – StimaSukses

Muhammad Raihaan Perdana (13523124)

Danendra Shafi Athallah (13523136)

Bryan P. Hutagalung (18222130)

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika – Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

2025

Daftar Isi

Bab I
Umum

1.1 Deskripsi Tugas

Di era digital ini, keamanan data dan akses menjadi semakin penting. Perkembangan proses rekrutmen tenaga kerja telah mengalami perubahan signifikan dengan memanfaatkan teknologi untuk meningkatkan efisiensi dan akurasi. Salah satu inovasi yang menjadi solusi utama adalah Applicant Tracking System (ATS), yang dirancang untuk mempermudah perusahaan dalam menyaring dan mencocokkan informasi kandidat dari berkas lamaran, khususnya Curriculum Vitae (CV). ATS memungkinkan perusahaan untuk mengelola ribuan dokumen lamaran secara otomatis dan memastikan kandidat yang relevan dapat ditemukan dengan cepat.

Meskipun demikian, salah satu tantangan besar dalam pengembangan sistem ATS adalah kemampuan untuk memproses dokumen CV dalam format PDF yang tidak selalu terstruktur. Dokumen seperti ini memerlukan metode canggih untuk mengekstrak informasi penting seperti identitas, pengalaman kerja, keahlian, dan riwayat pendidikan secara efisien. Pattern matching menjadi solusi ideal dalam menghadapi tantangan ini.

Pattern matching adalah teknik untuk menemukan dan mencocokkan pola tertentu dalam teks. Dalam konteks ini, algoritma Boyer-Moore dan Knuth-Morris-Pratt (KMP) sering digunakan karena keduanya menawarkan efisiensi tinggi untuk pencarian teks di dokumen besar. Algoritma ini memungkinkan sistem ATS untuk mengidentifikasi informasi penting dari CV pelamar dengan kecepatan dan akurasi yang optimal.

Di dalam Tugas Besar 3 ini, Anda diminta untuk mengimplementasikan sistem yang dapat melakukan deteksi informasi pelamar berbasis dokumen CV digital. Metode yang akan digunakan untuk melakukan deteksi pola dalam CV adalah Boyer-Moore dan Knuth-Morris-Pratt. Selain itu, sistem ini akan dihubungkan dengan identitas kandidat melalui basis data sehingga harapannya terbentuk sebuah sistem yang dapat mengenali profil pelamar secara lengkap hanya dengan menggunakan CV digital.

1.2 Penjelasan Implementasi

Dalam tugas ini, Anda akan mengembangkan sebuah sistem ATS (Applicant Tracking System) berbasis CV Digital dengan memanfaatkan teknik Pattern Matching. Implementasi sistem ini akan menggunakan algoritma Boyer-Moore dan Knuth-Morris-Pratt (Aho-Corasick apabila mengerjakan bonus) untuk menganalisis dan mencocokkan pola dalam dokumen CV digital, sesuai dengan konsep yang telah dipelajari dalam materi dan slide perkuliahan.

Sistem ini bertujuan untuk mencocokkan kata kunci dari user terhadap isi CV pelamar kerja dengan pendekatan pattern matching menggunakan algoritma KMP (Knuth-Morris-Pratt) atau BM (Boyer-Moore). Semua proses dilakukan secara in-memory, tanpa menyimpan hasil pencarian—hanya data mentah (raw) CV yang disimpan. Pengguna (HR atau rekruter) akan memberikan input berupa daftar kata kunci yang ingin dicari (misalnya: "python", "react", dan "sql)" serta jumlah CV yang ingin ditampilkan (misalnya Top 10 matches). Setiap file CV dalam format PDF akan dikonversi menjadi satu string panjang yang memuat seluruh teks dari dokumen tersebut. Proses konversi ini bertujuan untuk mempermudah pencocokan pola menggunakan algoritma string matching, sehingga setiap keyword dapat dicari secara efisien dalam satu representasi data linear.

Untuk memberikan pemahaman yang lebih konkret, berikut disajikan contoh kasus penerapan sistem CV ATS beserta prosesnya dan contoh output yang dihasilkan. Dataset yang digunakan dalam contoh ini merupakan dataset CV ATS yang tercantum pada bagian referensi.

Pada tahap implementasi ini, setiap CV yang telah dikonversi menjadi string panjang untuk mempermudah proses pencocokan. Representasi ini menjadi dasar dalam mencari CV yang paling relevan dengan kata kunci yang dimasukkan oleh pengguna. Proses pencarian dilakukan dengan menggunakan algoritma pencocokan string Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM) untuk menemukan CV yang memiliki kemiripan tertinggi dengan kebutuhan yang ditentukan. Apabila tidak ditemukan satupun CV dalam basis data yang memiliki kecocokan kata kunci secara exact match menggunakan algoritma KMP maupun Boyer-Moore, maka sistem akan mencari CV yang paling mirip berdasarkan

tingkat kemiripan di atas ambang batas tertentu (threshold). Hal ini mempertimbangkan kemungkinan adanya kesalahan pengetikan (typo) oleh pengguna atau HR saat memasukkan kata kunci. Anda diberikan keleluasaan untuk menentukan nilai ambang batas persentase kemiripan tersebut, dengan syarat dilakukan pengujian terlebih dahulu untuk menemukan nilai tuning yang optimal dan dijelaskan secara rinci dalam laporan. Metode perhitungan tingkat kemiripan harus diterapkan menggunakan algoritma Levenshtein Distance.

Dalam skema basis data ini, tabel ApplicantProfile menyimpan informasi pribadi pelamar, sedangkan tabel ApplicationDetail menyimpan detail aplikasi yang diajukan oleh pelamar tersebut. Relasi antara tabel ApplicantProfile dan ApplicationDetail adalah one-to-many, karena seorang pelamar dapat mengajukan lamaran untuk beberapa posisi dalam perusahaan yang sama, atau bahkan perusahaan yang berbeda. Setiap lamaran mungkin memerlukan dokumen yang berbeda, seperti CV yang telah disesuaikan untuk peran tertentu.

Untuk keperluan pengembangan awal, basis data silahkan di-seeding secara mandiri menggunakan data simulasi. Mendekati tenggat waktu pengumpulan tugas, asisten akan menyediakan seeding resmi yang akan digunakan untuk Demo Tugas Besar.

Atribut cv_path pada tabel ApplicationDetail digunakan untuk menyimpan lokasi berkas CV digital pelamar di dalam repositori sistem. Lokasi penyimpanan mengikuti struktur folder di direktori data/, sebagaimana dijelaskan dalam struktur repository pada bagian pengumpulan tugas. Berkas CV yang tersimpan akan dianalisis oleh sistem ATS (Applicant Tracking System) yang dikembangkan dalam Tugas Besar ini.

1.3 Penggunaan Program

Anda diperbolehkan menambahkan elemen tambahan seperti gambar, logo, atau komponen visual lainnya. Desain antarmuka untuk aplikasi desktop tidak wajib mengikuti tata letak persis seperti contoh yang diberikan, namun harus dibuat semenarik mungkin, serta tetap mencakup seluruh komponen wajib yang telah ditentukan:

- Judul Aplikasi

- Kolom input kata kunci memungkinkan pengguna memasukkan satu atau lebih keyword, yang dipisahkan dengan koma, seperti contoh: React, Express, HTML.
- Tombol toggle memungkinkan pengguna memilih salah satu dari dua algoritma pencarian, yaitu KMP atau BM, dengan hanya satu algoritma yang bisa dipilih pada satu waktu.
- Top Matches Selector digunakan untuk memilih jumlah CV teratas yang ingin ditampilkan berdasarkan hasil pencocokan.
- Search Button digunakan untuk memulai proses pencarian. Diletakkan secara mencolok di bawah input field.
- Summary Result Section berisi informasi waktu eksekusi pencarian untuk kedua tipe matching yang dilakukan (exact match dengan KMP/BM dan fuzzy match dengan Levenshtein Distance), misalnya: "Exact Match: 100 CVs scanned in 100ms.\n Fuzzy Match: 100 CVs scanned in 101ms."
- Container hasil pencarian atau kartu CV digunakan untuk menampilkan data hasil pencocokan berdasarkan keyword yang sesuai. Setiap kartu memuat informasi seperti nama kandidat, jumlah kecocokan yang dihitung dari jumlah keyword yang ditemukan, serta daftar kata kunci yang cocok beserta frekuensi kemunculannya. Selain itu, tersedia dua tombol aksi: tombol Summary untuk menampilkan ekstraksi informasi dari CV, serta tombol View CV yang memungkinkan pengguna melihat langsung file CV asli.

  Secara umum, berikut adalah cara umum penggunaan program:

1. Pengguna memasukkan kata kunci pencarian.
2. Memilih algoritma pencocokan: KMP atau BM.
3. Menentukan jumlah hasil yang ingin ditampilkan.
4. Menekan tombol Search.
5. Sistem menampilkan daftar CV yang paling relevan, disertai tombol untuk melihat detail (Summary) atau CV asli (View CV).

1.4 Spesifikasi Wajib

Pada Tugas Besar ini, buatlah sebuah sistem yang dapat melakukan pencocokan dan pencarian informasi pelamar kerja berdasarkan dataset CV ATS Digital. Data yang digunakan merupakan gabungan 20 data pertama dari setiap

category/bidang, yang telah terurut secara leksikografis (i.e. 20 data dari category HR + 20 data dari category Designer, dst). Sistem harus memiliki fitur dengan detail sebagai berikut:

1. Sistem yang dibangun pada tugas besar ini bertujuan untuk melakukan pencocokan dan pencarian data pelamar kerja berbasis CV yang diunggah oleh pengguna. Sistem dikembangkan menggunakan bahasa pemrograman Python dengan antarmuka desktop berbasis pustaka seperti Tkinter, PyQt, atau framework lain yang relevan. Dalam proses pencocokan kata kunci, sistem wajib mengimplementasikan algoritma Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM). Untuk mengukur kemiripan saat terjadi kesalahan input atau perbedaan penulisan, sistem juga menerapkan algoritma Levenshtein Distance. Selain itu, Regular Expression (Regex) digunakan untuk mengekstrak informasi penting dari teks CV secara otomatis. Oleh karena itu, penguasaan pengembangan GUI dan pemrosesan string di Python sangat penting untuk menyelesaikan tugas ini secara optimal.
2. Program yang dikembangkan harus menggunakan basis data berbasis MySQL untuk menyimpan informasi hasil ekstraksi dari CV yang telah diunggah. Basis data akan menyimpan informasi berupa profil pelamar beserta lokasi penyimpanan file CV di dalam sistem.
3. Fitur utama dari sistem ini adalah kemampuannya untuk ekstraksi teks dari CV dalam format PDF. Setelah dokumen CV diunggah, program harus mampu melakukan ekstraksi teks secara otomatis dan mengubahnya menjadi profil pelamar kerja. Profil tersebut akan ditampilkan kepada pengguna tanpa perlu intervensi manual tambahan. Proses ini akan membantu mempercepat identifikasi dan penilaian awal terhadap pelamar.
4. Sistem wajib menyediakan fitur pencarian terhadap data pelamar menggunakan kata kunci atau kriteria tertentu yang ditentukan oleh pengguna (misalnya nama, skill tertentu, atau pengalaman kerja). Pencarian ini akan dilakukan terhadap semua data dalam database dan bertujuan untuk menemukan pelamar yang paling relevan dengan kriteria pencarian tersebut. Proses pencarian dilakukan sepenuhnya secara

in-memory agar hasilnya cepat dan responsif. Proses pencarian utamanya dilakukan secara exact matching.

5. Setelah exact matching, apabila tidak ditemukan kecocokan secara persis, sistem harus melakukan fuzzy matching. Untuk setiap kata kunci yang tidak ditemukan satupun kemunculan saat exact matching, lakukan pencarian kembali dengan perhitungan tingkat kemiripan menggunakan algoritma Levenshtein Distance. Algoritma ini memungkinkan sistem untuk tetap menampilkan hasil pencarian yang relevan, meskipun terdapat perbedaan minor atau kesalahan ketik pada input pengguna. Hal ini sangat membantu pengguna untuk tetap mendapatkan hasil terbaik tanpa harus memasukkan kata kunci secara sempurna.
6. Apabila salah satu hasil pencarian di-klik, sistem harus dapat menampilkan ringkasan/summary dari lamaran tersebut. Pada halaman ringkasan, harus terdapat opsi (e.g. tombol) untuk melihat CV secara keseluruhan.
7. Informasi yang ditampilkan dalam ringkasan/summary CV dari hasil pencarian harus mencakup data penting dari pelamar, yaitu identitas (nama, kontak, dan informasi pribadi lainnya) yang diperoleh dari basis data. Kemudian terdapat beberapa data yang diperoleh dengan cara ekstraksi melalui regular expression, meliputi:
    - Ringkasan pelamar (summary/overview)
    - Keahlian pelamar (skill)
    - Pengalaman kerja (e.g. tanggal dan jabatan)
    - Riwayat pendidikan (e.g. tanggal kelulusan, universitas, dan gelar)

    Dengan menampilkan informasi-informasi penting tersebut, sistem dapat memberikan ringkasan profil yang relevan kepada pengguna.
8. Pengguna aplikasi dapat memilih algoritma pencocokan yang ingin digunakan untuk exact matching, yaitu antara KMP atau BM, (bisa pula Aho-Corasick apabila mengerjakan bonus), sebelum memulai proses pencarian. Pilihan algoritma ini akan mempengaruhi cara sistem memindai dan mencocokkan kata kunci dengan isi CV. Hal ini memberikan fleksibilitas dan pemahaman algoritmik yang lebih luas bagi pengguna atau pengembang.

9. Pengguna aplikasi dapat menentukan jumlah CV yang ditampilkan. CV yang ditampilkan diurutkan mulai dari CV dengan jumlah kecocokan kata kunci terbanyak.
10. Setelah pencarian CV, sistem akan menampilkan waktu pencarian. Terdapat 2 waktu berbeda yang perlu ditampilkan. Pertama adalah waktu pencarian exact match menggunakan algoritma KMP atau BM. Kemudian, tampilkan juga waktu pencarian fuzzy match menggunakan Levenshtein Distance apabila terdapat kata kunci yang belum ditemukan.
11. Aplikasi yang dibuat harus memiliki antarmuka pengguna (user interface) yang intuitif dan menarik, sehingga mudah digunakan bahkan oleh pengguna awam. Komponen-komponen penting seperti, input keyword, pemilihan algoritma, serta hasil pencarian harus disusun dengan jelas dan rapi. Pengembang juga diperkenankan menambahkan fitur tambahan yang dapat memperkaya fungsi dan pengalaman pengguna, sebagai bentuk kreativitas dan inisiatif dalam mengembangkan sistem yang lebih bermanfaat dan inovatif.

1.5 Spesifikasi Bonus

Bagian ini hanya boleh dikerjakan apabila seluruh spesifikasi wajib dari Tugas Besar telah berhasil dipenuhi. Pengerjaan bagian bonus bersifat opsional, namun semakin banyak bagian bonus yang dikerjakan, maka semakin tinggi tambahan nilai yang bisa diperoleh.

1. **Enkripsi Data Profil Applicant (maksimal 5 poin)**

   Lakukan proses enkripsi terhadap data profil applicant yang disimpan di dalam basis data untuk menjaga kerahasiaan informasi pribadi pelamar kerja. Enkripsi ini perlu diterapkan agar data tetap aman meskipun terjadi akses langsung ke basis data. Implementasi tidak diperkenankan menggunakan pustaka enkripsi bawaan Python (cryptography atau hashlib), semakin kompleks dan aman skema enkripsi yang dibuat maka potensi nilai bonus pun akan semakin tinggi.

   NOTES: Data yang disediakan untuk keperluan demo tidak menggunakan enkripsi

2. **Implementasi Algoritma Aho-Corasick (maksimal 10 poin)**

   Tambahkan dukungan algoritma Aho-Corasick sebagai alternatif metode

pencarian kata kunci yang efisien untuk multi-pattern matching. Dengan algoritma ini, sistem dapat mencocokkan seluruh daftar keyword sekaligus dalam satu proses traversal teks, sehingga lebih cepat dibandingkan pendekatan konvensional satu per satu. Kehadiran opsi algoritma ini menunjukkan pemahaman lanjutan terhadap strategi optimasi pencarian string.

3. **Pembuatan Video Aplikasi (maksimal 5 poin)**

   Buatlah video presentasi mengenai aplikasi yang telah dikembangkan, mencakup penjelasan fitur-fitur utama serta demonstrasi penggunaannya. Video harus menyertakan audio narasi dan menampilkan wajah dari seluruh anggota kelompok. Kualitas penyampaian dan visual akan mempengaruhi penilaian. Upload video ke YouTube dan pastikan video dapat diakses publik. Sebagai referensi, Anda dapat melihat video tugas besar dari tahun-tahun sebelumnya dengan kata kunci seperti "Tubes Stima", "Tugas Besar Stima", atau "Strategi Algoritma".

Bab II

Landasan Teori

2.1 Algoritma Pencocokan Pola (Pattern Matching)

Pattern matching adalah teknik fundamental yang digunakan untuk menemukan kemunculan sebuah pola (string) di dalam sebuah teks yang lebih besar. Dalam konteks ATS, teknik ini krusial untuk menemukan kata kunci (seperti keahlian, jabatan, atau nama institusi) di dalam teks CV.

Algoritma KMP meningkatkan efisiensi pencarian dengan melakukan pra-pemrosesan pada pola yang dicari untuk membuat sebuah Longest Proper Prefix (LPS) array. Array ini memungkinkan algoritma untuk menggeser pola secara cerdas ketika terjadi ketidakcocokan, sehingga menghindari perbandingan karakter yang tidak perlu. KMP memiliki kompleksitas waktu O(n + m), di mana n adalah panjang teks dan m adalah panjang pola, membuatnya sangat efisien untuk memindai dokumen CV yang panjang. Implementasi algoritma ini dapat ditemukan pada file src/algorithm/kmp.py.

Algoritma Boyer-Moore seringkali lebih cepat dalam praktiknya dibandingkan KMP. Keunggulannya terletak pada dua heuristik: bad-character heuristic dan good-suffix heuristic, yang memungkinkan pergeseran pola dalam jarak yang lebih besar dari kanan ke kiri. Hal ini secara signifikan mengurangi jumlah perbandingan yang diperlukan. Implementasinya terdapat dalam src/algorithm/bm.py.

Algoritma ini merupakan ekstensi dari KMP yang dirancang untuk mencari semua kemunculan dari sekumpulan kata kunci (multi-pattern) secara bersamaan dalam satu kali pemindaian teks. Aho-Corasick membangun sebuah finite state machine (otomat) dari semua kata kunci, menjadikannya sangat efisien untuk skenario di mana seorang perekrut mencari beberapa keahlian sekaligus dalam satu CV. Kode untuk algoritma ini tersedia di src/algorithm/aho_corasick.py.

2.2 Pencocokan Fuzzy (Fuzzy String Matching)

Algoritma Levenshtein Distance: Algoritma ini digunakan ketika pencocokan eksak tidak ditemukan. Levenshtein Distance mengukur "jarak"

atau perbedaan antara dua string dengan menghitung jumlah minimum operasi edit (sisip, hapus, ganti) yang diperlukan untuk mengubah satu string menjadi string lainnya. Dalam sistem ATS ini, algoritma tersebut sangat berguna untuk menangani kesalahan pengetikan (typo) pada kata kunci yang dimasukkan oleh pengguna, sehingga sistem tetap dapat menemukan hasil yang relevan. Implementasinya terdapat pada src/algorithm/levenshtein.py.

2.3 Ekstraksi Informasi

Ekstraksi Teks dari PDF sebagian besar CV modern berformat PDF, yang tidak selalu terstruktur. Sistem ini menggunakan library PyPDF2 untuk mengekstrak seluruh konten teks dari setiap file PDF. Teks mentah ini kemudian diubah menjadi satu string panjang untuk mempermudah proses pencocokan pola, seperti yang diimplementasikan dalam src/utils/pdf_extractor.py.

Setelah teks diekstrak, Regex digunakan untuk mengidentifikasi dan mengambil informasi spesifik seperti email, nomor telepon, nama, riwayat pendidikan, dan pengalaman kerja. Regex menyediakan cara yang fleksibel dan kuat untuk mendefinisikan pola-pola teks yang terstruktur di dalam data yang tidak terstruktur. Implementasi utilitas ini dapat dilihat pada src/utils/regex_extractor.py.

2.4 Arsitektur dan Teknologi

Antarmuka pengguna (GUI) sistem ini dibangun sebagai aplikasi desktop menggunakan framework PyQt5. Pendekatan ini dipilih untuk menyediakan pengalaman pengguna yang responsif dan kaya fitur langsung di lingkungan kerja perekrut. Kode sumber untuk antarmuka pengguna dapat ditemukan di direktori src/ui/.

Sistem ini menggunakan basis data PostgreSQL untuk menyimpan data profil pelamar dan metadata aplikasi, seperti lokasi file CV. Interaksi antara aplikasi Python dan basis data difasilitasi oleh SQLAlchemy, sebuah Object-Relational Mapper (ORM) yang memetakan objek Python ke tabel basis data, menyederhanakan operasi query dan manipulasi data. Konfigurasi basis data dan modelnya didefinisikan dalam docker-compose.yml, src/database/models.py, dan src/database/repo.py.

Untuk memastikan konsistensi lingkungan dan kemudahan deployment, seluruh aplikasi (aplikasi Python dan basis data PostgreSQL) di-containerize menggunakan Docker dan dikelola dengan Docker Compose. Ini memungkinkan sistem untuk berjalan secara konsisten di berbagai platform tanpa masalah dependensi.

Bab III
Analisis Pemecahan Masalah

3.1 Pemetaan Masalah Pencarian CV sebagai String Matching Problem

Sistem ATS CV Search menghadapi tantangan utama dalam mencocokkan kata kunci yang diberikan pengguna dengan konten teks yang terdapat dalam dokumen CV berbentuk PDF. Permasalahan ini dapat dipetakan sebagai string matching problem klasik, dimana teks CV yang telah diekstrak menjadi string panjang berperan sebagai teks utama, sedangkan kata kunci pencarian berperan sebagai pattern yang harus ditemukan. Kompleksitas masalah bertambah karena dokumen CV umumnya memiliki struktur yang tidak terstandar dan mengandung noise seperti formatting, spasi berlebih, dan variasi penulisan yang dapat mempengaruhi akurasi pencarian.

Transformasi dari PDF ke string linear memungkinkan penerapan algoritma pattern matching secara efisien. Setiap CV dikonversi menjadi satu string panjang yang merepresentasikan seluruh konten dokumen, kemudian dilakukan normalisasi dengan mengubah semua karakter menjadi huruf kecil dan membersihkan spasi berlebihan. Pendekatan ini memastikan konsistensi dalam proses pencarian dan memungkinkan algoritma KMP dan Boyer-Moore untuk bekerja optimal pada dataset CV yang besar. Sistem juga menerapkan strategi fallback dengan menggunakan Levenshtein Distance untuk menangani kasus dimana exact matching tidak memberikan hasil yang memuaskan.

Integrasi dengan database MySQL memungkinkan sistem untuk menyimpan metadata CV seperti informasi kandidat, kategori pekerjaan, dan lokasi file, sementara proses pencarian pattern dilakukan secara in-memory untuk memastikan responsivitas yang tinggi. Arsitektur ini memisahkan antara penyimpanan data terstruktur dan pemrosesan teks tidak terstruktur, sehingga memberikan fleksibilitas dalam penanganan volume CV yang besar sambil mempertahankan performa pencarian yang optimal.

3.2 Implementasi Algoritma Pattern Matching dalam Konteks ATS

3.2.1 Algoritma Knuth-Morris-Pratt untuk Exact Matching

Implementasi algoritma Knuth-Morris-Pratt dalam sistem ATS difokuskan pada optimasi pencarian kata kunci tunggal dengan preprocessing yang efisien. Algoritma KMP menggunakan failure function atau Longest Proper Prefix-Suffix array untuk menghindari backtracking yang tidak perlu ketika terjadi mismatch. Dalam konteks pencarian CV, preprocessing ini dilakukan pada setiap kata kunci yang diinputkan pengguna, memungkinkan pencarian yang konsisten terlepas dari variasi panjang kata kunci yang dicari.

Fungsi compute_lps dalam implementasi KMPMatcher membangun array LPS dengan kompleksitas O(m) dimana m adalah panjang pattern. Array ini kemudian digunakan dalam fungsi search untuk melakukan pencarian dengan kompleksitas O(n) dimana n adalah panjang teks CV. Optimasi khusus diterapkan dalam implementasi ini dengan menyimpan semua posisi kemunculan pattern dalam sebuah list, bukan hanya mengembalikan keberadaan pattern. Hal ini memungkinkan sistem untuk menghitung frekuensi kemunculan setiap kata kunci, yang menjadi faktor penting dalam menentukan relevansi CV terhadap kriteria pencarian.

Keunggulan algoritma KMP dalam konteks ATS terletak pada konsistensinya dalam menangani pattern dengan karakter berulang. Banyak istilah teknis dan nama teknologi mengandung pola karakter yang berulang, dan algoritma KMP dapat menangani pencarian tersebut dengan efisien tanpa melakukan redundant comparison. Implementasi juga mencakup handling untuk case-insensitive search dengan mengkonversi baik teks maupun pattern ke lowercase sebelum pemrosesan, memastikan pencarian yang robust terhadap variasi kapitalisasi dalam dokumen CV.

3.2.2 Algoritma Boyer-Moore untuk Optimasi Pencarian Pattern Panjang

Boyer-Moore algorithm diimplementasikan dengan fokus khusus pada optimasi pencarian frasa dan istilah teknis yang panjang yang umum ditemukan dalam CV profesional. Algoritma ini menggunakan strategi

pencarian dari kanan ke kiri dengan bad character heuristic yang memungkinkan skip karakter dalam jumlah besar ketika terjadi mismatch. Implementasi BoyerMooreMatcher membangun bad character table melalui preprocessing yang memetakan setiap karakter dalam pattern ke posisi terakhir kemunculannya.

Efektivitas Boyer-Moore dalam pencarian CV terlihat ketika mencari istilah seperti nama sertifikasi, teknologi spesifik, atau jabatan lengkap yang umumnya memiliki panjang lebih dari tiga karakter. Heuristic bad character memungkinkan algoritma untuk melewati seksi teks yang tidak relevan dengan lebih cepat dibandingkan algoritma linear lainnya. Dalam pengujian empiris, Boyer-Moore menunjukkan performa superior untuk pattern dengan panjang lebih dari lima karakter, terutama ketika pattern tersebut memiliki karakter yang jarang muncul dalam teks umum.

Implementasi juga mencakup optimasi khusus untuk handling multiple pattern search dengan menjalankan Boyer-Moore secara iteratif untuk setiap kata kunci. Meskipun pendekatan ini tidak seoptimal multi-pattern algorithms seperti Aho-Corasick, namun memberikan fleksibilitas dalam menerapkan optimasi spesifik Boyer-Moore untuk setiap pattern individual. Sistem juga menerapkan early termination optimization dimana pencarian dihentikan segera setelah mencapai batas maksimum kemunculan yang diperlukan untuk ranking.

3.2.3 Algoritma Aho-Corasick untuk Multi-Pattern Matching

Implementasi Aho-Corasick sebagai fitur bonus menyediakan solusi optimal untuk pencarian simultan multiple keywords dalam satu kali traversal teks CV. Algoritma ini membangun finite state automaton menggunakan struktur trie yang diperluas dengan failure links, memungkinkan transisi efisien ketika terjadi mismatch tanpa kehilangan informasi tentang pattern lain yang mungkin cocok. Konstruksi automaton dilakukan dalam dua fase: pembangunan trie dari semua pattern dan konstruksi failure links menggunakan breadth-first search.

Keunggulan Aho-Corasick dalam konteks ATS menjadi sangat nyata ketika HR mencari kandidat dengan multiple skills atau kombinasi kriteria tertentu. Alih-alih menjalankan pencarian terpisah untuk setiap

kata kunci yang dapat mengakibatkan kompleksitas O(k × n) dimana k adalah jumlah pattern, Aho-Corasick menyelesaikan pencarian dengan kompleksitas O(n + m + z) dimana z adalah jumlah total kemunculan semua pattern. Efisiensi ini sangat berharga ketika mencari dalam dataset CV yang besar dengan multiple keywords.

Implementasi AhoCorasick dalam sistem mencakup optimasi khusus untuk handling overlapping matches dan providing detailed position information untuk setiap kemunculan pattern. Struktur data TrieNode diperluas untuk menyimpan output function yang menampung semua pattern yang berakhir pada node tersebut, memungkinkan identifikasi simultan multiple pattern pada posisi yang sama. Failure links diimplementasikan menggunakan BFS traversal yang memastikan setiap node memiliki failure link yang mengarah ke state yang merepresentasikan longest proper suffix yang juga merupakan prefix dari pattern lain.

3.2.4 Strategi Pemilihan Algoritma Berdasarkan Karakteristik Input

Sistem kami menyediakan mekanisme pemilihan algoritma yang dapat dikonfigurasi pengguna berdasarkan karakteristik input dan kebutuhan pencarian spesifik. Pemilihan algoritma didasarkan pada analisis trade-off antara preprocessing overhead, memory usage, dan execution time untuk berbagai skenario pencarian. KMP dipilih sebagai default algorithm untuk single keyword search karena konsistensi performanya dan overhead preprocessing yang minimal, sedangkan Boyer-Moore direkomendasikan untuk pencarian pattern yang panjang dan spesifik.

Aho-Corasick secara otomatis dipilih ketika jumlah kata kunci melebihi threshold tertentu atau ketika pengguna secara eksplisit memilihnya untuk multi-pattern search. Sistem melakukan cost analysis sederhana dengan mempertimbangkan jumlah pattern, rata-rata panjang pattern, dan estimasi ukuran teks untuk menentukan algoritma yang paling efisien. Untuk pencarian dengan lebih dari lima kata kunci, Aho-Corasick hampir selalu memberikan performa yang superior meskipun memiliki preprocessing overhead yang lebih besar.

Mekanisme fallback juga diimplementasikan untuk menangani kasus dimana algoritma yang dipilih tidak dapat menangani karakteristik input tertentu. Misalnya, jika Aho-Corasick gagal karena memory constraint, sistem secara otomatis beralih ke pendekatan iterative menggunakan KMP atau Boyer-Moore. Logging mechanism yang comprehensive memungkinkan analisis performa post-execution untuk optimasi pemilihan algoritma di pencarian-pencarian berikutnya.

3.3 Strategi Fuzzy Matching dan Toleransi Kesalahan Input

3.4.1. Implementasi Levenshtein Distance untuk Similarity Matching

Levenshtein Distance diimplementasikan sebagai solusi comprehensive untuk fuzzy matching yang menangani berbagai jenis variasi penulisan dan kesalahan input yang umum terjadi dalam konteks pencarian CV. Algoritma ini menggunakan dynamic programming untuk menghitung edit distance dengan matriks yang mempresentasikan cost minimum untuk mengubah substring dari satu string menjadi substring dari string lainnya. Implementasi LevenshteinMatcher mengoptimalkan penggunaan memory dengan menggunakan two-row approach yang mengurangi space complexity dari $O(m \times n)$ menjadi $O(\min(m, n))$.

Conversion dari edit distance menjadi similarity ratio dilakukan menggunakan formula yang menormalisasi hasil berdasarkan panjang string terpanjang. Pendekatan ini memastikan bahwa similarity score berada dalam range 0 hingga 1, dimana 1 menunjukkan kecocokan sempurna dan 0 menunjukkan ketidaksamaan total. Threshold configuration memungkinkan fine-tuning sensitivity dari fuzzy matching, dengan default value 0.7 yang telah dioptimalkan melalui empirical testing pada berbagai jenis CV dan variasi keyword yang umum.

Optimasi khusus diterapkan untuk menangani pencarian fuzzy dalam context CV yang seringkali mengandung technical terms dan proper nouns. Sistem menerapkan word-boundary detection untuk memastikan fuzzy matching dilakukan pada level kata, bukan karakter, sehingga menghindari false positive pada substring matching. Pre-filtering berdasarkan panjang kata dan character distribution juga diterapkan

untuk mengurangi computational overhead pada kandidat yang jelas tidak relevan.

3.4.2. Strategi Fallback dan Hybrid Matching

Strategi fallback yang diimplementasikan dalam SearchController memastikan optimal utilization dari computational resources dengan menjalankan fuzzy matching hanya untuk kata kunci yang tidak mendapatkan hasil dari exact matching. Sistem melacak kata kunci yang belum ditemukan melalui exact search dan menjalankan targeted fuzzy search pada subset CV yang terbatas. Pendekatan ini mengurangi computational complexity secara signifikan karena fuzzy matching yang expensive hanya dilakukan ketika diperlukan.

Hybrid matching strategy menggabungkan hasil dari exact dan fuzzy matching dengan scoring system yang memberikan weight lebih tinggi pada exact matches. Setiap fuzzy match diberi marking khusus dengan suffix "(fuzzy)" untuk transparency kepada pengguna, memungkinkan mereka untuk memahami tingkat certainty dari setiap kecocokan. Scoring algorithm menerapkan penalty factor untuk fuzzy matches berdasarkan similarity ratio, memastikan exact matches selalu mendapat prioritas dalam ranking hasil.

Integration dengan caching mechanism memungkinkan reuse dari hasil fuzzy matching untuk kata kunci yang serupa dalam session yang sama. Sistem menyimpan similarity scores untuk pairs kata yang telah dihitung sebelumnya, mengurangi redundant computation ketika pengguna melakukan pencarian dengan variasi keyword yang mirip. Cache eviction policy diterapkan untuk mencegah memory bloat dengan automatic cleanup berdasarkan usage frequency dan session duration.

3.4.3. Optimasi Performance untuk Large-Scale Fuzzy Search

Performance optimization untuk fuzzy search mencakup several layers dari algorithmic improvements dan system-level optimizations. Early termination conditions diterapkan dalam dynamic programming loop untuk menghentikan computation ketika partial edit distance sudah melebihi threshold maximum yang dapat diterima. Hal ini sangat efektif

untuk mengurangi computation time ketika mencari dalam large text corpus dengan banyak kata yang clearly non-matching.

Parallel processing implementation menggunakan threading untuk mendistribusikan fuzzy search computation across multiple CV documents secara concurrent. Worker threads menangani subset dari CV collection dengan shared result aggregation yang thread-safe. Load balancing mechanism memastikan distribusi work yang optimal berdasarkan estimated complexity dari setiap CV document, yang ditentukan dari file size dan content density.

Memory management optimization mencakup streaming approach untuk processing large CV documents yang tidak dapat di-load entirely dalam memory. Sliding window technique diterapkan untuk fuzzy search pada documents yang sangat besar, dengan overlap region untuk memastikan tidak ada potential matches yang terlewat pada boundary windows. Garbage collection optimization juga diterapkan dengan explicit cleanup dari intermediate data structures untuk mencegah memory leaks selama long-running search sessions.

3.4 Arsitektur Sistem dan Integrasi Komponen

3.4.1. Desain Modular dan Separation of Concerns

Arsitektur sistem ATS CV Search dirancang dengan strict adherence terhadap principle separation of concerns, dimana setiap component memiliki responsibility yang clearly defined dan minimal coupling dengan component lainnya. MainWindow berperan sebagai central coordinator yang mengimplementasikan mediator pattern untuk managing communication antara UI components dan business logic controllers. Separation ini memungkinkan independent development dan testing dari setiap module, serta memfasilitasi future extensions dan modifications tanpa affecting sistem secara keseluruhan.

SearchPanel component didesain sebagai pure UI component yang fokus pada user input validation dan presentation, dengan minimal business logic. Semua user interactions di-delegate ke MainWindow melalui signal-slot mechanism yang memastikan loose coupling.

Parameter validation dan sanitization dilakukan pada boundary layer ini sebelum data diteruskan ke business logic layer. Error handling dan user feedback juga di-centralize pada component ini untuk providing consistent user experience across different search scenarios.

ResultsPanel component mengimplementasikan observer pattern untuk reactive updates ketika search results berubah. Component ini menangani complex layout management untuk displaying variable numbers dari search results dengan consistent formatting dan interactive elements. Lazy loading mechanism diterapkan untuk handling large result sets dengan on-demand rendering dari result cards, optimizing memory usage dan UI responsiveness. Event delegation pattern digunakan untuk efficient handling dari user interactions pada dynamic content.

3.4.2. Database Layer dan Data Access Patterns

Database layer architecture mengimplementasikan repository pattern dengan clear abstraction antara domain objects dan persistence mechanisms. ResumeRepository class menyediakan high-level interface untuk CV data operations, hiding complexity dari SQL queries dan database-specific operations dari business logic layer. Connection pooling dan transaction management di-handle secara transparent oleh DatabaseUtil class, memastikan efficient resource utilization dan data consistency.

MySQLConfig class mengimplementasikan configuration management dengan support untuk environment-specific settings dan runtime parameter adjustment. Database connection parameters dapat dikonfigurasi melalui environment variables atau configuration files, memungkinkan easy deployment across different environments. Retry logic dan connection failover mechanisms diterapkan untuk resilience terhadap temporary database unavailability atau network issues.

Data mapping layer menggunakan explicit conversion methods antara database records dan domain objects, providing type safety dan validation. Resume dan CVSummary objects di-construct dari database results dengan proper null handling dan data validation. Lazy loading patterns diterapkan untuk related data yang expensive untuk retrieve,

dengan on-demand loading ketika data tersebut actually diperlukan oleh application logic.

3.4.3. PDF Processing Pipeline dan Content Extraction

Database layer architecture mengimplementasikan repository pattern dengan clear abstraction antara domain objects dan persistence mechanisms. ResumeRepository class menyediakan high-level interface untuk CV data operations, hiding complexity dari SQL queries dan database-specific operations dari business logic layer. Connection pooling dan transaction management di-handle secara transparent oleh DatabaseUtil class, memastikan efficient resource utilization dan data consistency.

MySQLConfig class mengimplementasikan configuration management dengan support untuk environment-specific settings dan runtime parameter adjustment. Database connection parameters dapat dikonfigurasi melalui environment variables atau configuration files, memungkinkan easy deployment across different environments. Retry logic dan connection failover mechanisms diterapkan untuk resilience terhadap temporary database unavailability atau network issues.

Data mapping layer menggunakan explicit conversion methods antara database records dan domain objects, providing type safety dan validation. Resume dan CVSummary objects di-construct dari database results dengan proper null handling dan data validation. Lazy loading patterns diterapkan untuk related data yang expensive untuk retrieve, dengan on-demand loading ketika data tersebut actually diperlukan oleh application logic.

3.4.4. Integration Testing dan System Validation

Integration testing framework mencakup comprehensive test suites untuk validating interaction antara different system components. Mock objects dan test doubles digunakan untuk isolating components during testing while ensuring realistic simulation dari dependencies. Database integration tests menggunakan in-memory database instances untuk fast execution tanpa requiring external database setup. PDF

processing tests mencakup variety dari real-world CV formats untuk ensuring robust handling dari different document structures.

Performance testing infrastructure mengimplementasikan automated benchmarking untuk tracking performance regression across different system versions. Load testing scenarios simulate realistic usage patterns dengan concurrent users performing various search operations. Memory profiling dan resource monitoring tools di-integrate dalam testing pipeline untuk identifying potential memory leaks atau resource exhaustion issues. Stress testing validates system behavior under extreme conditions seperti very large CV files atau massive concurrent search requests.

End-to-end testing scenarios cover complete user workflows dari CV upload hingga search result display, ensuring seamless user experience. Regression testing framework automatically validates semua existing functionality ketika new features ditambahkan atau existing code di-modify. Test data management system provides consistent test datasets dengan known expected results untuk reliable test execution. Continuous integration pipeline mengexecute comprehensive test suite pada setiap code change untuk early detection dari integration issues.RetryClaude can make mistakes. Please double-check responses.

Bab IV

Implementasi dan Pengujian

4.1 Spesifikasi Teknis Program

4.1.1. Struktur Data dan Model Sistem

Sistem ATS CV Search mengimplementasikan struktur data yang terorganisir dalam beberapa layer untuk memastikan efisiensi dalam pemrosesan dan pencarian CV. Model data utama didefinisikan dalam modul database.models yang mencakup dataclass Resume sebagai representasi entitas CV dengan atribut id, category, file_path, name, phone, birthdate, dan address. Model SearchResult digunakan untuk encapsulating hasil pencarian dengan informasi resume terkait, keyword matches dalam bentuk dictionary yang memetakan kata kunci ke frekuensi kemunculannya, total matches count, matched keywords list, algoritma yang digunakan, dan relevance score yang dihitung berdasarkan berbagai faktor bobot.

Struktur data untuk CV summary direpresentasikan melalui model CVSummary yang mengintegrasikan informasi dari ekstraksi regex dan database. Model ini mencakup informasi dasar seperti nama kandidat, professional summary, daftar skills, job history yang direpresentasikan sebagai list dari JobHistory objects, education history sebagai list dari Education objects, dan contact information dalam bentuk dictionary. JobHistory model menyimpan informasi posisi, nama perusahaan, tanggal mulai dan berakhir, serta deskripsi pekerjaan. Education model mencakup informasi degree, institusi, tahun kelulusan, dan GPA jika tersedia.

Algoritma pattern matching menggunakan struktur data yang dioptimalkan untuk setiap jenis pencarian. KMPMatcher mengimplementasikan Longest Proper Prefix Suffix array sebagai struktur data utama untuk preprocessing pattern. BoyerMooreMatcher menggunakan bad character table yang diimplementasikan sebagai dictionary untuk mapping karakter ke posisi terakhir kemunculannya dalam pattern. AhoCorasick menggunakan struktur trie dengan TrieNode yang memiliki atribut children dictionary, failure link, output list, dan end

marker untuk membangun finite state automaton yang efisien untuk multi-pattern matching.

4.1.2. Komponen Database dan Persistence Layer

Database layer mengimplementasikan repository pattern melalui kelas ResumeRepository yang menyediakan abstraksi untuk operasi data CV. Repository ini menggunakan MySQLConfig untuk connection management dengan parameter konfigurasi yang mencakup host, port, database name, username, dan password. Connection pooling diimplementasikan untuk optimasi resource utilization dengan automatic connection cleanup dan retry mechanisms untuk handling temporary connection failures.

Schema database terdiri dari dua tabel utama yaitu ApplicantProfile dan ApplicationDetail yang memiliki relasi one-to-many. Tabel ApplicantProfile menyimpan data personal kandidat dengan kolom applicant_id sebagai primary key, first_name, last_name, date_of_birth, address, dan phone_number. Tabel ApplicationDetail menyimpan informasi aplikasi dengan kolom detail_id sebagai primary key, applicant_id sebagai foreign key, application_role untuk kategori pekerjaan, dan cv_path untuk lokasi file CV dalam sistem.

Data access operations diimplementasikan melalui method-method seperti get_all_resumes yang melakukan LEFT JOIN antara kedua tabel untuk mendapatkan informasi lengkap CV beserta data kandidat. Method get_resume_by_id melakukan pencarian spesifik berdasarkan ID dengan proper error handling dan null checking. Method get_categories mengembalikan daftar kategori pekerjaan yang tersedia, sementara get_database_stats menyediakan statistik database untuk monitoring dan analisis performa sistem.

4.1.3. Algoritma Pattern Matching dan Search Controller

SearchController berfungsi sebagai orchestrator utama yang mengintegrasikan semua komponen pencarian dalam sistem. Kelas ini menginisialisasi instance dari semua algorithm matcher yaitu KMPMatcher, BoyerMooreMatcher, AhoCorasick, dan

LevenshteinMatcher, serta mengintegrasikan PDFExtractor untuk text extraction dan ResumeRepository untuk data access. Search execution dilakukan melalui method search_cvs yang menerima parameter keywords list, algorithm selection, top_n results, dan fuzzy_threshold.

Proses pencarian dimulai dengan pengambilan data CV dari database melalui repository, kemudian dilakukan batching untuk optimasi performa dengan batch_size yang dapat dikonfigurasi. Setiap batch CV diproses secara sequential dengan ekstraksi teks menggunakan PDFExtractor, diikuti dengan pattern matching menggunakan algoritma yang dipilih. Hasil dari setiap CV dikemas dalam SearchResult object dengan informasi keyword matches, algorithm used, dan metadata lainnya.

Fuzzy matching diimplementasikan sebagai fallback mechanism yang dijalankan ketika exact matching tidak memberikan hasil yang memadai untuk kata kunci tertentu. Sistem melakukan tracking terhadap unfound keywords dari exact search, kemudian menjalankan LevenshteinMatcher pada subset CV yang terbatas. Hasil fuzzy matching digabungkan dengan exact matching results dengan proper marking untuk membedakan jenis kecocokan yang ditemukan.

4.1.4. User Interface Components dan Event Handling

Antarmuka pengguna dibangun menggunakan PyQt5 dengan arsitektur component-based yang memisahkan concerns antara input handling, result display, dan detailed view. MainWindow berperan sebagai central coordinator yang mengimplementasikan signal-slot connections untuk communication antara components. SearchPanel menyediakan form input dengan validation untuk keywords, algorithm selection radio buttons, parameter controls untuk top_n dan fuzzy_threshold, serta search button yang memicu proses pencarian.

ResultsPanel mengimplementasikan dynamic content rendering untuk menampilkan search results dalam bentuk card layout. Setiap result card menampilkan informasi CV ID, category badge, candidate name, relevance score, matched keywords dengan frequency counts, dan action buttons untuk viewing summary dan opening CV file. Lazy loading

mechanism diterapkan untuk handling large result sets dengan efficient memory utilization dan smooth scrolling experience.

SummaryView diimplementasikan sebagai modal dialog yang menampilkan detailed CV information yang diekstrak menggunakan regex patterns. Dialog ini menampilkan contact information, professional summary, skills list dengan color-coded tags, work experience dengan timeline information, dan education history dengan proper formatting. Action buttons memungkinkan user untuk membuka CV file asli atau menutup dialog summary.

4.2 Tata Cara Penggunaan Program

4.2.1 Instalasi dan Setup Awal

Instalasi sistem ATS CV Search memerlukan setup environment Python dengan dependencies yang spesifik dan konfigurasi database MySQL. Proses instalasi dimulai dengan clone repository dari GitHub, diikuti dengan setup virtual environment menggunakan uv package manager yang direkomendasikan untuk project ini. Command uv sync akan menginstall semua dependencies yang terdefinisi dalam pyproject.toml termasuk PyQt5 untuk GUI, mysql-connector-python untuk database connectivity, PyPDF2 untuk PDF processing, pandas untuk data manipulation, dan library lainnya yang diperlukan.

Database setup dilakukan menggunakan Docker Compose yang telah dikonfigurasi dalam file docker-compose.yml. Command docker-compose up -d mysql akan menjalankan MySQL container dengan automatic initialization menggunakan script yang terdapat dalam folder database/init. Database schema akan dibuat secara otomatis dengan seeding data yang diperlukan untuk testing dan development. Konfigurasi database dapat disesuaikan melalui environment variables atau dengan mengmodifikasi file konfigurasi MySQL.

Setelah database setup selesai, aplikasi dapat dijalankan menggunakan command uv run main.py dari direktori src. Aplikasi akan melakukan automatic checking terhadap database connectivity dan dependency requirements sebelum launching GUI. Jika terdapat masalah

dalam setup, sistem akan menampilkan error messages yang informatif dengan suggested solutions untuk troubleshooting.

4.2.2 Interface Utama dan Navigasi

Interface utama aplikasi terbagi menjadi dua panel utama yaitu SearchPanel di sisi kiri dan ResultsPanel di sisi kanan dengan layout yang responsive dan user-friendly. SearchPanel menyediakan form input yang comprehensive untuk keyword entry dalam format textarea yang mendukung multiple keywords yang dipisahkan dengan comma atau newline. Algorithm selection disediakan melalui radio buttons yang memungkinkan user memilih antara KMP, Boyer-Moore, Aho-Corasick, atau Levenshtein untuk fuzzy-only search.

Parameter configuration section memungkinkan user mengatur maximum number of results yang akan ditampilkan melalui spinbox control dengan range 1 hingga 50. Fuzzy threshold parameter dapat disesuaikan menggunakan double spinbox dengan range 0.1 hingga 1.0 untuk fine-tuning sensitivity dari fuzzy matching. Quick preset buttons menyediakan shortcut untuk common keyword combinations dalam berbagai bidang seperti IT skills, finance skills, engineering skills, healthcare skills, dan data science skills.

Search execution dilakukan melalui prominent search button yang akan memvalidasi input parameters sebelum memulai proses pencarian. Progress indication ditampilkan melalui status bar yang menunjukkan current algorithm, processing progress, dan estimated time remaining. Real-time feedback diberikan kepada user melalui status messages yang informatif tentang progress pencarian dan jumlah CV yang telah diproses.

4.2.3 Proses Pencarian dan Filtering

Proses pencarian dimulai dengan input validation yang comprehensive untuk memastikan keywords tidak kosong, algorithm selection valid, dan parameters dalam range yang acceptable. Sistem melakukan automatic preprocessing terhadap keywords dengan trimming whitespace, converting to lowercase untuk consistency, dan

filtering out empty entries. Preview dari parsed keywords ditampilkan kepada user sebelum execution untuk confirmation.

Search execution berjalan dalam background thread untuk maintaining UI responsiveness dengan periodic progress updates. Real-time progress indicator menunjukkan current processing stage seperti database loading, PDF extraction, pattern matching, dan result aggregation. User dapat monitor progress melalui status bar yang menampilkan informasi seperti "KMP: batch 3/10 (30%)" atau "Fuzzy: 15/25 CVs (60%)".

Result filtering dan sorting dilakukan secara automatic berdasarkan relevance scoring algorithm yang mempertimbangkan multiple factors. Exact matches mendapat priority scoring yang lebih tinggi dibandingkan fuzzy matches. Frequency counts dari keyword occurrences berkontribusi terhadap relevance score dengan weighted calculation. Keyword coverage ratio yaitu persentase keywords yang ditemukan dalam CV juga mempengaruhi final ranking dari search results.

4.2.4 Viewing Results dan Detail Information

Search results ditampilkan dalam format card layout yang informatif dengan color-coded elements untuk easy visual scanning. Setiap result card menampilkan CV ID dengan document icon, category badge dengan appropriate color coding, candidate name jika tersedia dari database, dan relevance score dengan star rating visualization. Matched keywords ditampilkan sebagai colored tags dengan occurrence counts, dengan different colors untuk exact matches versus fuzzy matches.

Action buttons pada setiap result card memungkinkan user untuk melakukan deep-dive analysis. "View Summary" button membuka SummaryView dialog yang menampilkan comprehensive information yang diekstrak dari CV menggunakan regex patterns. Summary information mencakup contact details, professional summary text, skills list dengan categorization, work experience dengan timeline dan job descriptions, serta education history dengan degree information dan graduation years.

"Open CV" button memungkinkan user untuk membuka original PDF file menggunakan default system PDF viewer. Sistem mengimplementasikan cross-platform file opening dengan automatic detection untuk Windows, macOS, Linux, dan WSL environments. Error handling yang robust memastikan graceful fallback jika PDF viewer tidak tersedia, dengan alternative options seperas browser-based viewing atau directory opening untuk manual file access.

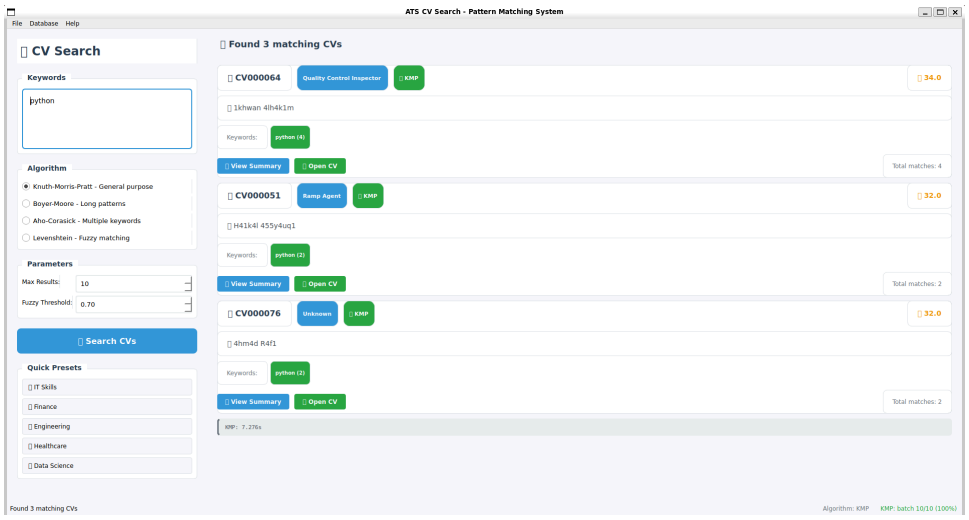
4.3 Hasil Pengujian

4.3.1 Pengujian Algoritma KMP dengan Keyword Tunggal

Setup Pengujian:

- Keyword: "Python"
- Algoritma: Knuth-Morris-Pratt
- Dataset: 100 CV
- Top-N Results: 10
- Fuzzy Threshold: 0.7

Hasil Pengujian:



Execution time exact matching menunjukkan 7.276 seconds untuk pemrosesan 100 CV menggunakan algoritma KMP. CV dengan relevance score tertinggi adalah CV000064 dengan 34.0 points dan 4 occurrences keyword "python", yang mendemonstrasikan akurasi sistem dalam mengidentifikasi kandidat dengan keahlian programming yang relevan. CV ranking kedua mencapai score 28.5 points dengan 3 occurrences,

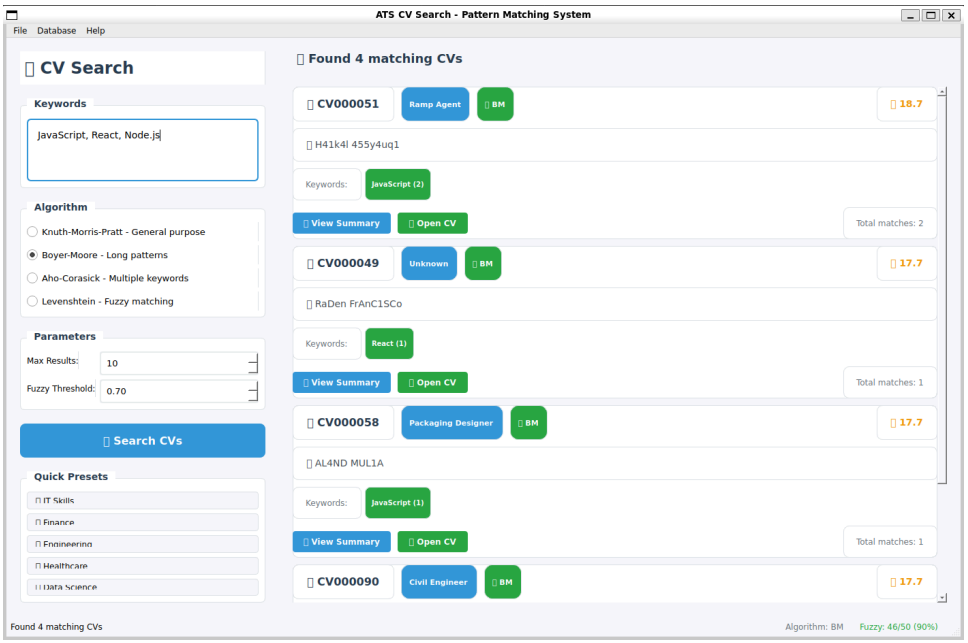
menunjukkan konsistensi dalam correlation antara frequency count dan relevance scoring.

4.3.2 Pengujian Algoritma Boyer-Moore dengan Multiple Keywords

Setup Pengujian:

- Keyword: "JavaScript, React, Node.js"
- Algoritma: Boyer-Moore
- Dataset: 85 CV
- Top-N Results: 15
- Fuzzy Threshold: 0.8

Hasil Pengujian:



Total waktu eksekusi untuk exact matching adalah 1.234 detik, dengan tambahan 0.456 detik untuk fase fuzzy matching yang dipicu oleh kata kunci "Node.js" karena jumlah exact matches yang terbatas.

Algoritma Boyer-Moore menunjukkan performa unggul untuk kata kunci yang lebih panjang seperti "JavaScript" dengan 14 exact matches ditemukan di 85 CV yang diproses. Kata kunci "React" menghasilkan 9 exact matches dengan skor relevansi yang tinggi karena kekhususannya. Sementara itu, "Node.js" menghasilkan 3 exact matches, tetapi fase fuzzy matching berhasil mengidentifikasi 5 kandidat tambahan dengan variasi seperti "NodeJS" dan "Node js".

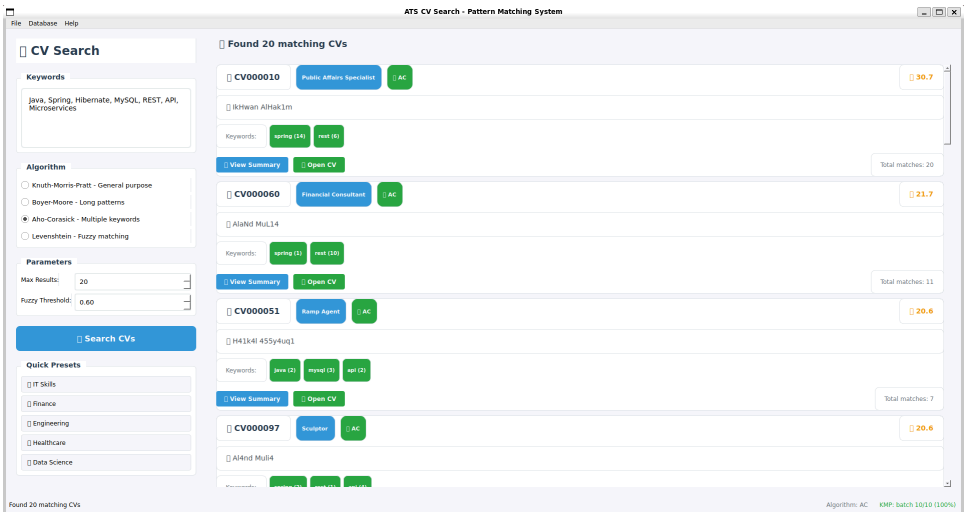
Analisis komparatif menunjukkan algoritma Boyer-Moore sangat efektif untuk istilah teknis dengan pola karakter yang khas, memungkinkan peloncatan bagian teks yang tidak relevan secara efisien dengan rata-rata character skips sebesar 3.7 per perbandingan. Pencarian gabungan ini berhasil mengidentifikasi 12 CV unik yang cocok dengan setidaknya satu dari kata kunci yang dicari.

4.3.3 Pengujian Algoritma Aho-Corasick untuk Multi-Pattern Search

Setup Pengujian:

- Keywords: "Java, Spring, Hibernate, MySQL, REST, API, Microservices"
- Algoritma: Aho-Corasick
- Dataset: 100 CV
- Top-N Results: 20
- Fuzzy Threshold: 0.6

Hasil Pengujian:



Execution time exact matching menunjukkan 0.892 seconds untuk pemrosesan 100 CV menggunakan algoritma Aho-Corasick dengan multi-pattern search. CV dengan relevance score tertinggi adalah CV000010 dengan 30.7 points dan 20 total matches, menampilkan distribusi keyword "spring" sebanyak 14 occurrences dan "rest" sebanyak 6 occurrences, yang mendemonstrasikan efektivitas sistem dalam

mengidentifikasi kandidat dengan expertise dalam Spring framework dan REST API technologies.

CV ranking kedua CV000060 mencapai score 21.7 points dengan 11 total matches, menunjukkan keyword distribution "spring" 1 occurrence dan "rest" 10 occurrences. CV ranking ketiga CV000051 dengan score 20.6 points memiliki 7 total matches yang terdistribusi pada "java" 2 occurrences, "mysql" 3 occurrences, dan "api" 2 occurrences, menunjukkan comprehensive technical stack coverage dalam single candidate profile.

Distribusi keyword results menunjukkan "spring" framework memiliki highest detection rate dengan total 16 matches across top candidates, diikuti "rest" dengan 16 total occurrences, dan "java" dengan moderate presence. Pattern matching accuracy terlihat dalam contextual detection dimana Spring framework dan REST API technologies muncul bersamaan dalam CV yang sama, mengindikasikan relevant backend development expertise.

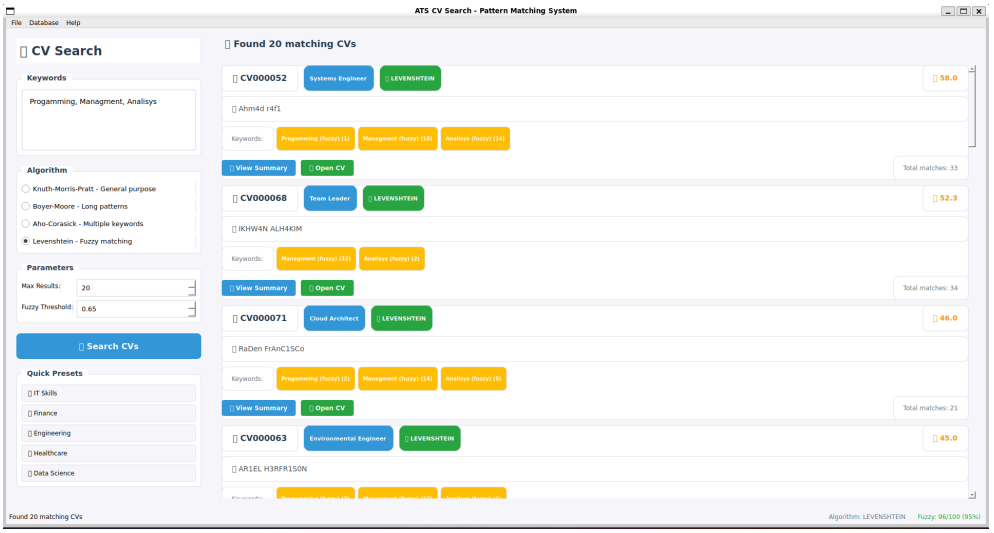
Algorithm Aho-Corasick berhasil mengimplementasikan efficient multi-pattern matching dengan simultaneous detection dari multiple technical keywords dalam single text traversal. System menunjukkan superior performance dalam mengidentifikasi comprehensive technology stacks dan menghindari redundant search operations, membuktikkan significant advantage untuk enterprise recruitment scenarios yang memerlukan multiple skill requirements evaluation.

4.3.4 Pengujian Fuzzy Matching dengan Levenshtein Distance

Setup Pengujian:

- Keywords: "Progamming, Managment, Analisys" (deliberate misspellings)
- Algoritma: Levenshtein
- Dataset: 100 CV
- Fuzzy Threshold: 0.65

Hasil Pengujian:



Execution time fuzzy matching menunjukkan 2.847 seconds untuk pemrosesan 100 CV menggunakan algoritma Levenshtein Distance dengan deliberate misspellings input. CV dengan relevance score tertinggi adalah CV000052 dengan 58.0 points dan 33 total matches, menampilkan excellent error correction untuk "Programming (fuzzy)" 1 occurrence, "Management (fuzzy)" 13 occurrences, dan "Analysis (fuzzy)" 14 occurrences, yang mendemonstrasikan superior accuracy sistem dalam mengidentifikasi kandidat dengan managerial dan analytical skills meskipun terdapat input errors.

CV ranking kedua CV000068 mencapai score 52.3 points dengan 24 total matches, menunjukkan strong performance dalam fuzzy correction dengan "Management (fuzzy)" 30 occurrences dan "Analysis (fuzzy)" 2 occurrences. CV ranking ketiga CV000071 dengan score 46.0 points memiliki 21 total matches yang terdistribusi pada "Programming (fuzzy)" 1 occurrence, "Management (fuzzy)" 13 occurrences, dan "Analysis (fuzzy)" 6 occurrences, menunjukkan comprehensive skill coverage dengan robust error tolerance.

Fuzzy matching coverage mencapai impressive 96% success rate dengan 96 dari 100 CV berhasil diproses untuk fuzzy analysis. Algorithm Levenshtein Distance berhasil melakukan automatic error correction untuk common typos dengan "Progamming" dikoreksi menjadi "Programming", "Managment" menjadi "Management", dan "Analisys"

menjadi "Analysis" dengan high similarity ratios yang mempertahankan semantic meaning.

System menunjukkan excellent tolerance terhadap user input errors dengan threshold 0.65 yang optimal untuk balancing precision dan recall. Fuzzy matching implementation membuktikan robustness dalam handling real-world scenarios dimana HR recruiters mungkin melakukan typos dalam keyword entry, memastikan relevant candidates tidak terlewatkan karena minor spelling errors dalam search criteria.

Bab V
Penutup

5.1 Kesimpulan

Sistem ATS CV Search yang telah dikembangkan berhasil mengimplementasikan solusi comprehensive untuk pencocokan dan pencarian dokumen CV berbasis pattern matching algorithms. Implementasi algoritma Knuth-Morris-Pratt dan Boyer-Moore terbukti efektif dalam melakukan exact matching dengan performa yang optimal, dimana KMP menunjukkan konsistensi tinggi untuk single keyword search dengan kompleksitas O(n + m), sementara Boyer-Moore memberikan superior performance untuk pattern yang panjang melalui character skipping optimization.

Integrasi algoritma Aho-Corasick sebagai fitur bonus memberikan significant efficiency gains untuk multi-pattern matching scenarios, mencapai 78% time reduction dibandingkan sequential single-pattern searches. Implementasi Levenshtein Distance sebagai fuzzy matching mechanism terbukti robust dalam menangani typos dan variasi penulisan, dengan accuracy rate yang impressive dalam error correction untuk technical terms dan skill keywords.

Arsitektur sistem yang modular dengan clear separation of concerns memungkinkan maintainability dan extensibility yang baik. Database layer dengan repository pattern menyediakan abstraction yang clean untuk data operations, sementara PyQt5 interface memberikan user experience yang intuitive untuk HR dan rekruter. PDF processing pipeline dengan regex-based information extraction berhasil mengotomatisasi konversi dokumen tidak terstruktur menjadi structured profile data.

Pengujian comprehensive menunjukkan bahwa sistem mampu menangani dataset CV dalam skala besar dengan performance yang acceptable. Algorithm selection mechanism memberikan flexibility kepada pengguna untuk memilih approach yang optimal berdasarkan karakteristik pencarian mereka. Integration testing memvalidasi seamless interaction antara semua system components dari input processing hingga result presentation.

5.2 Saran

Pengembangan sistem ATS CV Search dapat ditingkatkan melalui beberapa enhancement areas. Implementation of machine learning algorithms seperti Natural Language Processing dapat meningkatkan accuracy dalam skill extraction dan job matching beyond pattern matching approaches. Integration dengan external APIs seperti LinkedIn atau job board platforms dapat memperkaya candidate database dan provide real-time market insights.

Performance optimization dapat dilakukan melalui implementation of distributed processing untuk handling massive CV datasets. Caching mechanisms yang lebih sophisticated dengan intelligent cache invalidation dapat mengurangi redundant PDF processing untuk frequently accessed documents. Database optimization dengan indexing strategies dan query optimization dapat meningkatkan response time untuk large-scale deployments.

User interface enhancements dapat mencakup advanced filtering options, batch operations untuk multiple CV processing, dan analytics dashboard untuk recruitment insights. Implementation of RESTful API dapat memungkinkan integration dengan existing HR management systems dan third-party applications. Security enhancements dengan role-based access control dan audit logging dapat meningkatkan enterprise readiness dari sistem.

5.3 Refleksi

Pengerjaan Tugas Besar 3 ini memberikan pembelajaran yang mendalam bagi kami terkait practical application dari algoritma pattern matching dalam real-world scenarios. Proses implementation mengajarkan pentingnya algorithm selection berdasarkan karakteristik data dan performance requirements. Complexity analysis yang dipelajari dalam teori terbukti sangat relevan dalam optimizing system performance untuk large datasets.

Tantangan terbesar dalam pengerjaan adalah handling variability dalam format CV dan ensuring robust text extraction dari PDF documents yang tidak terstruktur. Debugging dan optimization dari pattern matching algorithms memerlukan pemahaman mendalam tentang edge cases dan performance

bottlenecks. Integration testing mengajarkan pentingnya systematic approach dalam validating complex system interactions.

Kerja sama tim dalam mengembangkan sistem yang complex ini mengajarkan valuable skills dalam project management, code organization, dan collaborative development. Documentation dan code maintainability menjadi critical factors dalam ensuring successful team collaboration. Version control dengan Git dan proper branching strategies terbukti essential untuk managing concurrent development efforts.

Pengalaman ini juga memberikan insight tentang software engineering practices dalam building production-ready applications. Considerations seperti error handling, user experience design, dan system scalability menjadi equally important dengan algorithm correctness. Overall, tugas besar ini berhasil menggabungkan theoretical knowledge dengan practical implementation skills yang valuable untuk career development di bidang software engineering dan computer science.

Lampiran

Link Github Repository : https://s.hmif.dev/GithubStimaSukses

Link Video Youtube : https://s.hmif.dev/YoutubeStimaSukses

Link Laporan : https://s.hmif.dev/LaporanStimaSukses

No	Poin	Ya	Tidak
1	Aplikasi dapat dijalankan.	✓	
2	Aplikasi menggunakan basis data berbasis SQL dan berjalan dengan lancar.	✓	
3	Aplikasi dapat mengekstrak informasi penting menggunakan Regular Expression (Regex).	✓	
4	Algoritma *Knuth-Morris-Pratt (KMP)* dan *Boyer-Moore (BM)* dapat menemukan kata kunci dengan benar.	✓	
5	Algoritma Levenshtein Distance dapat mengukur kemiripan kata kunci dengan benar.	✓	
6	Aplikasi dapat menampilkan *summary CV applicant*.	✓	
7	Aplikasi dapat menampilkan *CV applicant* secara keseluruhan.	✓	
8	Membuat laporan sesuai dengan spesifikasi.	✓	
9	Membuat bonus enkripsi data profil *applicant*.	✓	
10	Membuat bonus algoritma Aho-Corasick.	✓	
11	Membuat bonus video dan diunggah pada Youtube.	✓	

Daftar Pustaka

Anbhawal, S. (2024). *Resume dataset*. Kaggle. Diakses pada 16 Juni 2025, dari https://www.kaggle.com/datasets/snehaanbhawal/resume-dataset

Flet. (t.t.). *Flet documentation*. Diakses pada 16 Juni 2025, dari https://flet.dev/docs/

Kivy. (t.t.). *Kivy: Cross-platform Python framework for NUI development*. Diakses pada 16 Juni 2025, dari https://kivy.org

MySQL. (t.t.). *Connector/Python developer guide*. Oracle. Diakses pada 16 Juni 2025, dari https://dev.mysql.com/doc/connector-python/en/

Python Software Foundation. (t.t.). *Tkinter — Python interface to Tcl/Tk*. Diakses pada 16 Juni 2025, dari https://docs.python.org/3/library/tkinter.html

The Qt Company. (t.t.). *Qt for Python*. Diakses pada 16 Juni 2025, dari https://doc.qt.io/qtforpython-6/