

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

Laporan Tugas Besar 3
IF2211 Strategi Algoritma



Disusun oleh Kelompok 10 (juliusev):

13523025 Joel Hotlan Haris Siahaan

13523030 Julius Arthur

13622076 Ziyen Agil Nur Ramadhan

**Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung**

2025

DAFTAR ISI

DAFTAR ISI.....	1
Bab 1 Deskripsi Tugas.....	2
Bab 2 Landasan Teori.....	7
a. Dasar Teori.....	7
b. Penjelasan Aplikasi.....	9
Bab 3 Analisis Pemecahan Masalah.....	11
a. Langkah-Langkah dan Proses Pemetaan Masalah menjadi Algoritma KMP dan BM.....	11
b. Fitur Fungsional dan Arsitektur Aplikasi yang Dibangun.....	12
Bab 4 Implementasi dan Pengujian.....	14
a. Spesifikasi Teknis Program.....	14
b. Tata Cara Penggunaan Program.....	17
c. Analisis dan Hasil Pengujian.....	18
Bab 5 Kesimpulan dan Saran.....	21
a. Kesimpulan.....	21
b. Saran.....	21
LAMPIRAN.....	22
DAFTAR PUSTAKA.....	23

Bab 1

Deskripsi Tugas



Gambar 1. CV ATS dalam Dunia Kerja
(sumber: <https://www.antaraneews.com/>)

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.

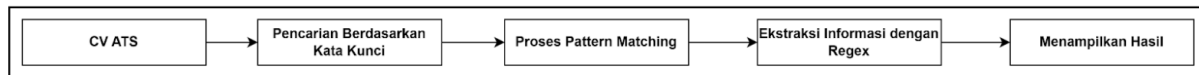
Pattern matching adalah teknik untuk menemukan dan mencocokkan pola tertentu dalam teks. Dalam konteks ini, algoritma Boyer-Moore dan Knuth-Morris-Pratt (KMP) sering digunakan karena keduanya menawarkan efisiensi tinggi untuk pencarian teks di dokumen besar. Algoritma ini memungkinkan sistem ATS untuk 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.

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.




Gambar 2. Skema Implementasi Applicant Tracking System

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.

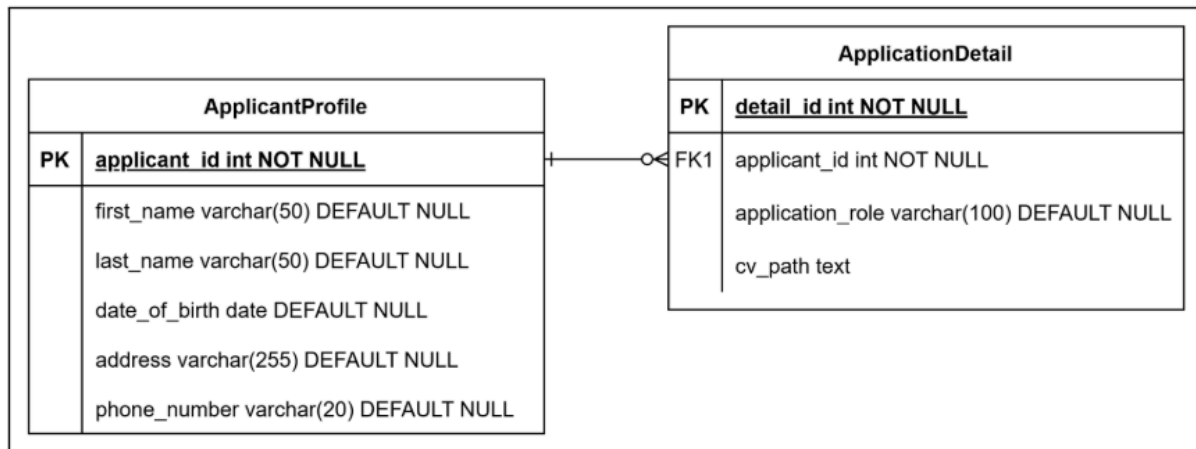
Untuk memberikan pemahaman yang lebih konkret, berikut disajikan contoh kasus penerapan sistem CV ATS beserta prosesnya dan contoh output yang dihasilkan. Dataset yang digunakan dalam contoh ini merupakan dataset CV ATS yang tercantum pada bagian referensi.

Tabel 1. Hasil ekstraksi teks dari CV ATS

CV ATS	Ekstraksi Text untuk Regex	Ekstraksi Text untuk <i>Pattern Matching</i> (KMP & BM)
 10276858.pdf	Ekstraksi Text Regex.txt	Ekstraksi Text Pattern Matching.txt

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.



Gambar 3. Skema Basis Data CV ATS

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.

Penggunaan Program

CV Analyzer App

Keywords :
React, Express, HTML

Search Algorithm:
KMP ☒ BM

Top Matches:
3

Search

Results
100 CVs scanned in 100ms

Farhan 4 matches
Matched keywords:
1. React: 1 occurrence
2. Express: 2 occurrences
3. HTML: 1 occurrence
Summary View CV

Aland 1 match
Matched keywords:
1. React: 1 occurrence
Summary View CV

Ariel 1 match
Matched keywords:
1. Express: 1 occurrence
Summary View CV

Gambar 4. Contoh Antarmuka Program (Halaman Home)

CV Summary

Farhan
Birthdate: 05-19-2025
Address: Masjid Salman ITB
Phone: 0812 3456 7890

Skills:
React Express HTML

Job History:
CTO
2003-2004
Leading the organization's technology strategies

Education:
Informatics Engineering (Institut Teknologi Bandung)
2022-2026

Gambar 5. Contoh Antarmuka Program (Halaman Summary)

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).

Bab 2

Landasan Teori

a. Dasar Teori

Pencocokan string adalah proses mencari kemunculan pola (*pattern*) string dalam suatu teks yang memiliki string lebih panjang dari pola string tersebut. Misalkan terdapat suatu teks dan pola berisi string berikut.

Teks: `abaabxabaaba`

Pola: `abaaba`

Pencocokan string akan mencari apakah terdapat pola '`abaaba`' dalam teks yang diberikan. Terdapat beberapa algoritma yang dapat digunakan untuk mencari kemunculan pola dalam teks seperti, algoritma KMP (Knuth-Morris-Pratt), algoritma BM (Boyer-Moore), algoritma Aho-Corasick, algoritma Levenshtein Distance, serta *regular expression* (regex).

(1) Algoritma KMP

Algoritma ini dilakukan dengan membangun sebuah tabel untuk menghitung fungsi pinggiran. Fungsi pinggiran ini akan menyimpan informasi tentang prefiks terpanjang dari pola yang juga merupakan sufiks dari setiap prefiks pola. Tabel ini akan digunakan sebagai acuan untuk menentukan banyaknya pergeseran pola jika ditemukan ketidakcocokan. Berikut adalah ilustrasi fungsi pinggiran.

Tabel 2. Ilustrasi Fungsi Pinggiran

j	0	1	2	3	4	5
P[j]	a	b	a	a	b	a
k		0	1	2	3	4
b(k)		0	0	1	1	2

Fungsi pinggiran dihitung dengan mencari prefiks dan sufiks terpanjang yang dapat dibentuk dari suatu posisi pada pola. Panjang maksimal string untuk membentuk prefiks dan sufiks pada suatu pola adalah satu kurangnya dari panjang string yang sedang ditinjau pada posisi tertentu. Misalkan jika ingin meninjau fungsi pinggiran pada posisi ke-3 ($j = 2$), maka panjang maksimal prefiks dan sufiksnya adalah 2. Berikut adalah penjelasan untuk tabel di atas.

(a) $j = 0$

Karena pada posisi ini tidak mungkin membentuk suatu prefiks dan sufiks, maka nilai fungsi pinggirannya adalah 0.

(b) $j = 1$

Pada posisi ini, prefiks yang dapat dibentuk adalah {a} sedangkan sufiks yang dapat dibentuk adalah {b}. Karena tidak ada kecocokan antara prefiks dan sufiks, maka nilai fungsi pinggirannya adalah 0.

(c) $j = 2$

Pada posisi ini, prefiks yang dapat dibentuk adalah {a, ab} sedangkan sufiks yang dapat dibentuk adalah {a, ba}. Kecocokan antara prefiks dan sufiks ditemukan pada string a sehingga nilai fungsi pinggirannya adalah 1 karena panjang dari string a adalah 1.

(d) $j = 3$

Pada posisi ini, prefiks yang dapat dibentuk adalah {a, ab, aba} sedangkan sufiks yang dapat dibentuk adalah {a, aa, baa}. Kecocokan antara prefiks dan sufiks ditemukan pada string a sehingga nilai fungsi pinggirannya adalah 1 karena panjang dari string a adalah 1.

(e) $j = 4$

Pada posisi ini, prefiks yang dapat dibentuk adalah {a, ab, aba, abaa} sedangkan sufiks yang dapat dibentuk adalah {b, ab, aab, baab}. Kecocokan antara prefiks dan sufiks ditemukan pada string ab sehingga nilai fungsi pinggirannya adalah 2 karena panjang dari string ab adalah 2.

Setelah mendapatkan nilai dari fungsi pinggiran, algoritma KMP akan menggunakannya untuk menentukan banyaknya pergeseran pola jika ditemukan ketidakcocokan. Jika ketidakcocokan ditemukan pada string ke- j pada pola, maka pergeseran pola dilakukan sebanyak $j-b(k)$.

(2) Algoritma Boyer-Moore

Jika algoritma KMP mencocokkan string dari kiri ke kanan, algoritma BM melakukan pencocokan string dimulai dari kanan ke kiri. Terdapat tiga kasus yang akan dijumpai jika menggunakan algoritma ini. Berikut adalah ilustrasinya.

(a) Kasus 1

0	1	2	3	4	5	6	7	8	9	10	11	12
a	b	a	a	b	x	?	?	?	?	?	?	?
x	b	a	x	b	a							
		x	b	a	x	b	a					

Pada contoh di atas, string pola langsung mengalami ketidakcocokan pada pencocokan pertama (*bad-character rule*). String pada pola menunjukkan huruf a dan string pada teks menunjukkan huruf x. Algoritma BM akan menyimpan string pada teks saat ini (dalam hal ini adalah x) dan mencari apakah terdapat string x pada pola. Kasus 1 menunjukkan terdapat string x pada pola (untuk hasil yang tidak menunjukkan terdapat string x pada pola akan dibahas di kasus 3). Dengan begitu, pola akan digeser sedemikian sehingga x pada pola dan teks saling berimpit. Jika terdapat lebih dari satu buah x pada pola, maka string x yang dipilih untuk pergeseran adalah string x paling kanan.

(Untuk hasil yang menunjukkan tidak terdapat string x pada pola akan dibahas di kasus 3)

(b) Kasus 2

0	1	2	3	4	5	6	7	8	9	10	11	12
a	b	a	x	a	x	?	?	?	?	?	?	?
c	b	c	a	a	x							
	c	b	c	a	a	x						

Pada contoh di atas, string mengalami ketidakcocokan pada indeks ke-3 (pencocokkan ke-3). Kasus ini akan menggunakan *good-suffix rule* di mana algoritma akan mencari apakah terdapat string sufiks pada pola. Pada kasus di atas, sufiks yang terbentuk adalah ax dan akan dicek apakah terdapat ax di sebelah kiri string ketidakcocokan. Jika ada, string akan digeser hingga string sufiks pada pola dan teks berimpit sedangkan jika tidak ada, string akan digeser sebanyak 1 ke kanan.

(c) Kasus 3

0	1	2	3	4	5	6	7	8	9	10	11	12
a	b	a	a	b	x	c	?	?	?	?	?	?
a	b	a	a	b	a	c						
						a	b	a	a	b	a	c

Pada contoh di atas, string mengalami ketidakcocokan pada indeks ke-5 (pencocokkan ke-2). Kasus ini merupakan alternatif lain untuk kasus 1 jika tidak ditemukan string x pada pola. Pada kasus ini, algoritma akan menggeser pola sedemikian sehingga string pertama pada pola (indeks ke-0) berada di posisi satu lebihnya dari posisi string teks yang memiliki ketidakcocokan (dalam kasus di atas, pola digeser hingga indeks pertamanya di posisi ke-6 karena memiliki ketidakcocokan di indeks ke-5).

b. Penjelasan Aplikasi

Tugas besar ini akan mengimplementasikan kedua algoritma di atas untuk membangun sebuah aplikasi ATS (*Application Tracking System*) berbasis CV Digital. Aplikasi dibangun dengan bahasa pemrograman Python dengan antarmuka desktop berbasis pustaka seperti Tkinter, PyQt, dan framework lainnya. Komponen aplikasi ini terdiri atas.

- (1) Judul Aplikasi: (JULIUSCV).
- (2) Kolom input kata kunci yang memungkinkan pengguna memasukkan satu atau lebih kata kunci yang ingin dicari. Kata kunci ini dipisahkan dengan koma, seperti contoh: React, Express, HTML.
- (3) 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.

- (4) *Top Matches Selector* digunakan untuk memilih jumlah CV teratas yang ingin ditampilkan berdasarkan hasil pencocokan.
- (5) Tombol pencarian digunakan untuk memulai proses pencarian.
- (6) *Summary Result Section* berisi informasi waktu eksekusi pencarian untuk kedua tipe pencocokan yang dilakukan (exact match dengan KMP/BM dan fuzzy match dengan Levenshtein Distance).
- (7) Kontainer hasil pencarian atau kartu CV yang digunakan untuk menampilkan data hasil pencocokan berdasarkan kata kunci yang sesuai. Setiap kartu memuat informasi seperti nama kandidat, jumlah kecocokan yang dihitung dari jumlah kata kunci yang ditemukan, serta daftar kata kunci yang cocok beserta frekuensi kemunculannya. Selain itu, tersedia dua tombol aksi yaitu tombol Summary untuk menampilkan ekstraksi informasi dari CV, serta tombol View CV yang memungkinkan pengguna melihat langsung file CV asli.

Bab 3

Analisis Pemecahan Masalah

a. Langkah-Langkah dan Proses Pemetaan Masalah menjadi Algoritma KMP dan BM

Proses pemetaan masalah:

- **Teks**

Dalam masalah ini, teks adalah string panjang hasil ekstraksi pdf CV menjadi string panjang lalu melalui proses yang mengubah setiap karakter menjadi huruf kecil, menghapus karakter khusus, dan menghapus spasi ganda untuk memudahkan pencocokan string.

- **Pattern**

Dalam masalah ini, pattern adalah pola-pola yang diterima dari masukkan pengguna untuk di cocokan dengan file-file CV yang ada.

Langkah-langkah pemecahan masalah :

1. Ekstraksi Path CV
Mengambil daftar path file CV dari basis data.
2. Ekstraksi Teks PDF
Mengubah file PDF menjadi string teks menggunakan pustaka pdfplumber.
3. Normalisasi Teks
Mengubah teks menjadi huruf kecil, menghapus karakter khusus, dan menghapus spasi ganda supaya pencocokan lebih mudah diimplementasikan.
4. Input Kata Kunci
Mengambil kata kunci dari pengguna dan memecahnya menjadi list kata kunci.
5. Pencocokan Exact
Melakukan pencarian setiap kata kunci pada teks menggunakan algoritma yang dipilih yaitu antara Knuth-Morris-Pratt atau Boyer-Moore.
6. Penggunaan Fuzzy matching
Jika tidak ada teks yang memiliki pola kecocokan exact, maka sistem akan membandingkan pola dengan potongan teks dengan menggunakan Levenshtein Distance dengan toleransi tertentu.
7. Penilaian Skor dan Peringkat Hasil
Mengurutkan hasil berdasarkan jumlah kecocokan tertinggi untuk ditampilkan sebagai hasil pencarian.
8. Menampilkan Hasil
Hasil ditampilkan dalam bentuk kartu CV dengan skor kecocokan, keyword, dan aksi untuk melihat ringkasan atau file CV.

b. Fitur Fungsional dan Arsitektur Aplikasi yang Dibangun

Aplikasi ini memiliki beberapa fitur-fitur fungsional sebagai berikut :

- **Fitur Ekstraksi Teks dari File PDF**

Aplikasi secara otomatis membaca file PDF dari path yang tersimpan di database pelamar. Seluruh isi CV dikonversi menjadi string teks biasa supaya dapat dianalisis oleh algoritma string matching.

- **Fitur Pencarian Keyword**

Aplikasi memungkinkan pengguna untuk memasukkan satu atau lebih keyword yang ingin di cari dari daftar CV pelamar dengan setiap keyword dipisahkan tanda koma. Fitur ini dapat digunakan dengan memasukkan keyword di kolom pencarian.

- **Fitur Memilih Algoritma**

Aplikasi memungkinkan pengguna untuk menentukan algoritma pencarian. Algoritma pencarian yang diimplementasikan adalah Knuth-Morris-Pratt dan Boyer-Moore. Fitur ini diimplementasikan dalam bentuk tombol toggle dengan dua pilihan yaitu KMP dan BM.

- **Fitur Fuzzy Matching**

Aplikasi memungkinkan pencocokan kata kunci dengan Levenshtein Distance untuk mencari kata yang mirip. Hal ini berguna untuk menangani kesalahan ketik atau variasi penulisan sebuah kata.

- **Fitur Perhitungan dan Penyajian Hasil Pencarian**

Aplikasi menampilkan CV hasil pencarian berdasarkan jumlah kemunculan kata kunci dan tingkat kemiripan. Hasil ditampilkan dalam bentuk kartu CV yang berisi nama pelamar, jumlah kata kunci yang cocok, serta kata kunci yang ditemukan dan jumlah kemunculannya . Kata kunci yang ditemukan dan jumlah kemunculan, serta tombol aksi untuk melihat ringkasan dan melihat CV.

- **Fitur Ringkasan Profil**

Aplikasi memungkinkan pengguna untuk melihat ringkasan profil seorang pelamar yang CV nya mengandung pola seperti keyword. Fitur ini diimplementasikan dengan tombol view summary.

- **Fitur Membuka File CV**

Aplikasi memungkinkan pengguna untuk membuka seorang pelamar yang CV nya mengandung pola seperti keyword. Fitur ini diimplementasikan dengan tombol view summary.

- **Arsitektur Aplikasi**

Arsitektur dari aplikasi Applicant Tracking System ini adalah sebagai berikut.

1. Tampilan Utama

- 1.1. Judul Aplikasi

- 1.2. Tagline Aplikasi

- 1.3. Radio Button Toggle Pemilihan Algoritma
 - 1.4. Kolom Input Jumlah Hasil Maksimum
 - 1.5. Tombol Eksekusi Pencarian
2. Tampilan Hasil
 - 2.1. Statistik Pencarian
 - 2.2. Kartu Hasil CV
 - 2.2.1. Nama Pelamar
 - 2.2.2. Jumlah Kecocokan
 - 2.2.3. Kata kunci yang ditemukan dan jumlah kemunculannya
 - 2.2.4. Tombol aksi
 - 2.2.4.1. Tombol Lihat Ringkasan
 - 2.2.4.2. Tombol Lihat CV
3. Tampilan Ringkasan
 - 3.1. Identitas Pelamar
 - 3.1.1. Nama Pelamar
 - 3.1.2. Tanggal Lahir
 - 3.1.3. Alamat
 - 3.1.4. Nomor Telepon
 - 3.2. Daftar Keterampilan
 - 3.3. Daftar Pendidikan
 - 3.4. Riwayat Pekerjaan
 - 3.5. Navigasi Modal
 - 3.5.1. tombol Back
 - 3.5.2. tombol View CV

Bab 4

Implementasi dan Pengujian

a. Spesifikasi Teknis Program

- **Kelas**

Kelas	Keterangan
PDFExtractor	Berisi fungsi-fungsi yang melakukan ekstraksi plaintext dari PDF normalisasi teks.
CVInfo	Kelas ini berfungsi sebagai data container untuk menyimpan hasil ekstraksi informasi dari file CV dalam bentuk terstruktur; title, skills, ringkasan, pencapaian, pengalaman kerja, riwayat pendidikan, dan teks mentah.
CVRegexExtractor	Berisi fungsi dan prosedur yang melakukan ekstraksi informasi penting dari dokumen CV berbasis teks hasil konversi dari file PDF dengan teknik regular expression(regex).
IntegratedCVProcessor	Berisi gabungan fungsionalitas utama dari kelas PDFTextExtractor dan CVRegexExtractor.
Backend	Berfungsi sebagai penghubung logika backend(python) dengan antarmuka frontend HTML yang ditampilkan dengan PyQt.
JuliusCVApp	Kelas utama untuk membangun tampilan GUI aplikasi berbasis PyQt5.

- **Fungsi**

- File kmp.py

Fungsi	Keterangan
compute_lps(pattern:str) -> List[int]	Menghitung tabel LPS(Longest Prefix which also Suffix) untuk sebuah pola.
kmp_search(pattern:str, text:str) -> int	Mencari semua kemunculan pattern dalam text dan mengembalikan jumlah kemunculannya.

- File boyer_moore.py

Fungsi	Keterangan
boyer_moore_search(text:str, pattern:str)	Mencari semua kemunculan pattern dalam teks dengan memanggil fungsi boyer_moore() dan mengembalikan semua indeksinya.
boyer_moore(text:str,	Mencari kemunculan pertama pola dalam teks dengan

pattern:str)	algoritma Boyer-Moore.
--------------	------------------------

- Kelas PDFExtractor

Fungsi	Keterangan
extract_text(self, pdf_path: str, clean: bool = True)	Mengekstrak teks dari seluruh halaman pada file PDF dan menggabungkannya menjadi string panjang serta melakukan normalisasi tergantung parameter clean.
clean_text(self, text: str)	Membersihkan teks ekstraksi dari PDF supaya siap diproses oleh algoritma string matching.

- Kelas CVInfo

Fungsi	Keterangan
__post_init__(self)	Metode khusus dari dataclasses Python yang menginisiasi objek setelah CVInfo dibuat.

- Kelas CVRegexExtractor

Fungsi	Keterangan
__init__(self)	Fungsi konstruktor yang menjalankan setup_patterns() saat objek dibuat.
setup_patterns(self)	Menyiapkan pola regex untuk mendeteksi berbagai bagian CV serta mendefinisikan pattern untuk judul dan header section (skills, summary, highlights, accomplishments, experience, education).
find_section_boundaries(self, text: str)	Mencari semua header section dan posisinya dalam teks serta mengembalikan dictionary dengan nama section dan posisi baris.
extract_section_content_flexible(self, text: str, section_name: str)	Mengekstrak semua konten dari section tertentu, menggunakan batas section yang ditemukan oleh find_section_boundaries(), serta mengembalikan list berisi semua baris dalam section tersebut.
extract_title(self, text: str)	Mengekstrak nama atau job title dari awal CV, mencari dalam 10 baris pertama dan mencocokkan dengan pattern title, serta menghindari header section yang terdeteksi sebagai title.
extract_experience_flexible(self, text: str)	Mengekstrak semua konten pengalaman kerja.
extract_education_flexible(self, text: str)	Mengekstrak semua konten pendidikan.

<code>extract_education(self, text: str)</code>	Mengekstrak semua konten pendidikan (mendeteksi degree, major, year, university, dan location).
<code>read_extracted_text(self, file_path: str)</code>	Membaca file teks yang sudah diekstrak serta menangani error jika file tidak ditemukan.
<code>extract_cv_info(self, file_path: str)</code>	Fungsi utama yang menggabungkan semua ekstraksi, membaca file, mengekstrak semua bagian CV, dan mengembalikan objek CVInfo.
<code>format_output(self, cv_info: CVInfo)</code>	Memformat informasi CV yang sudah diekstrak menjadi string yang rapi, menambahkan bullet points dan spacing yang sesuai.
<code>save_formatted_output(self, cv_info: CVInfo, output_path: str)</code>	Menyimpan hasil format ke file output, menggunakan <code>format_output()</code> untuk mendapatkan teks terformat.

- Kelas IntegratedCVProcessor

Fungsi	Keterangan
<code>__init__(self)</code>	Menginisialisasi objek dari PDFExtractor dan CVRegexExtractor.
<code>process_pdf(pdf_path: str, output_dir: str = None)</code>	Mengekstrak teks dari file PDF dan mengambil informasi-informasi penting dari teks CV.
<code>process_multiple_pdfs(pdf_directory: str, output_dir: str = None)</code>	Melakukan ekstraksi dari banyak PDF dalam satu folder sekaligus.
<code>print_extraction_summary(cv_info: CVInfo)</code>	Mencetak ringkasan informasi hasil ekstraksi dari suatu CV.

- Kelas Backend

Fungsi	Keterangan
<code>__init__(self)</code>	Membuat instance dari IntegratedCVProcessor yang akan digunakan untuk mengekstrak teks dari file CV.
<code>searchCVs</code>	Mencari CV berdasarkan kata kunci menggunakan algoritma pencocokan teks seperti KMP dan Boyer-Moore.
<code>openFile(path)</code>	Membuka file CV langsung menggunakan aplikasi default OS.
<code>getCVSummary(cv_path)</code>	Mengambil ringkasan informasi dari suatu CV

	berdasarkan pathnya.
--	----------------------

- Kelas JuliusCVApp

Fungsi	Keterangan
<code>__init__(self)</code>	Menginisialisasi GUI aplikasi dengan membuat <code>QWebEngineView</code> , menghubungkan frontend dengan backend serta memuat isi HTML gui.
<code>load_html_content(self)</code>	Membaca file html gui dan menampilkannya ke <code>QWebEngine</code> .

- File LevenshteinDistance.py

Fungsi	Keterangan
<code>LevenshteinDistance(source: str, target: str)</code>	Menghitung Levenshtein Distance untuk string source menjadi target dan mengembalikan hasil dalam bentuk matriks.
<code>fuzzy_search(text: str, pattern: str, max_distance=2)</code>	Melakukan fuzzy matching terhadap pola dalam teks menggunakan Levenshtein Distance, dengan default jarak maksimum 2.

- File db.py

Fungsi	Keterangan
<code>get_db_connection()</code>	Mengembalikan objek koneksi ke database untuk digunakan oleh fungsi-fungsi lain.
<code>create_db()</code>	Menginisialisasi sistem basis data untuk menyimpan tabel <code>ApplicantProfile</code> dan <code>ApplicationDetail</code> serta memasukkan data ke tabel.
<code>get_paths()</code>	Mengambil semua path file CV yang ada di basis data beserta identitasnya.
<code>get_applicant_profile_by_id(path)</code>	Mengambil informasi rinci pelamar berdasarkan path CV-nya.

b. Tata Cara Penggunaan Program

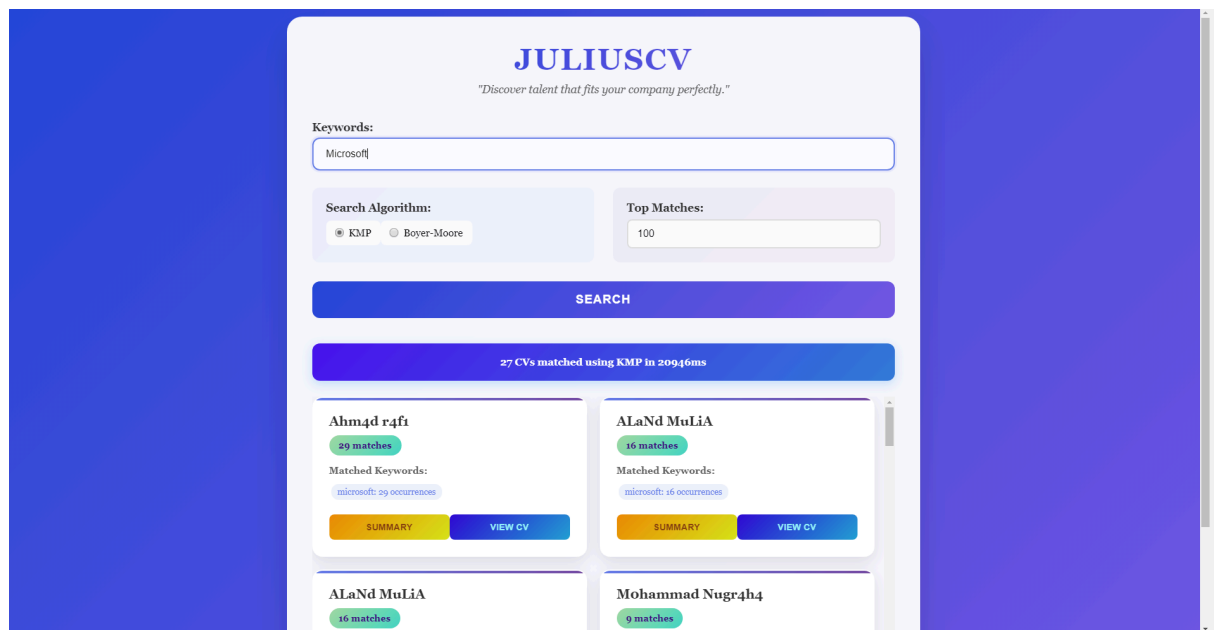
1. Pengguna memasukkan informasi database di dalam file `.env`.
2. Pengguna mengunduh seluruh *dependencies* yang telah dicatat pada file `requirements.txt`.
3. Pengguna menjalankan program dengan `src` sebagai *current directory*.

4. Dalam aplikasi, pengguna dapat memasukkan kata kunci yang ingin dicari. Jika terdapat lebih dari satu kata kunci, pengguna memasukkannya dengan koma sebagai pemisah.
5. Pengguna memilih algoritma *pattern matching* serta jumlah CV yang ingin ditampilkan.
6. Pengguna dapat melihat ringkasan dan membuka file pdf CV melalui hasil query.

c. Analisis dan Hasil Pengujian.

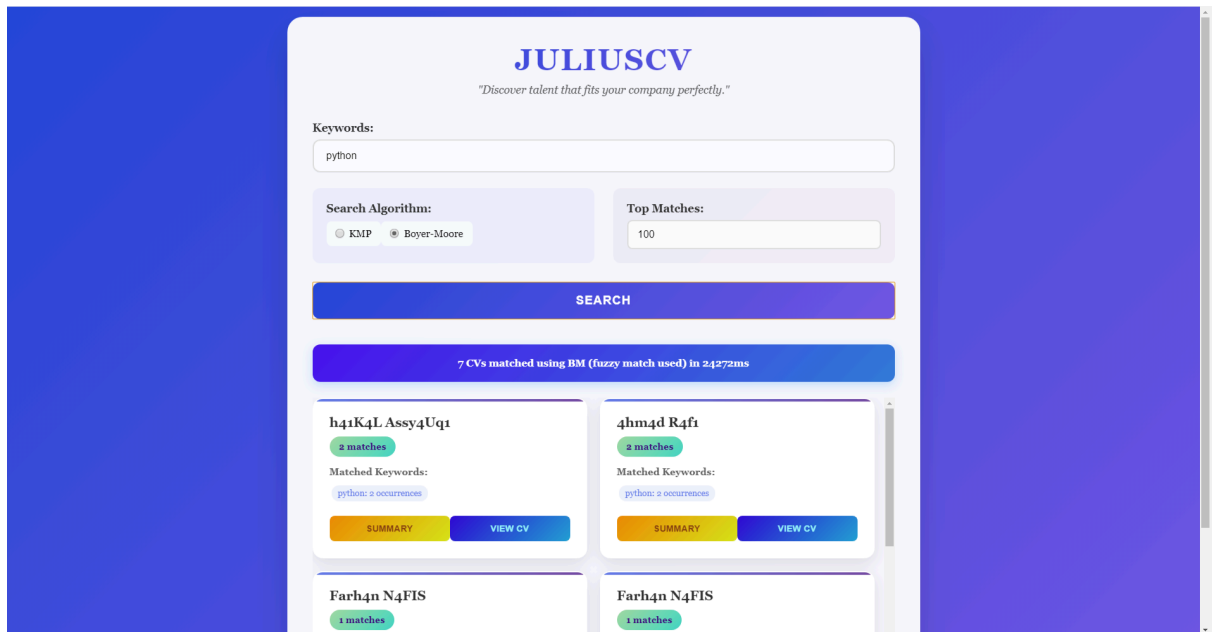
Pengujian dilakukan menggunakan data pada folder INFORMATION-TECHNOLOGY dari sample resmi yang diberikan asisten.

1. Keyword: Microsoft
Algoritma: KMP



Program berhasil menemukan CV yang memiliki kata kunci Microsoft. Dengan menggunakan algoritma KMP, berhasil didapatkan 27 CV dengan kata Microsoft dalam waktu sekitar 20 detik.

2. Keyword: Python
Algoritma: Boyer-Moore



Berdasarkan hasil pengujian antarmuka aplikasi, sistem berhasil menampilkan hasil pencarian CV berdasarkan kata kunci yang dimasukkan pengguna. Pada pengujian ini, kata kunci "python" dicari dengan algoritma Boyer-Moore dan batas atas pencarian ditetapkan sebanyak 100 CV. Sistem menunjukkan bahwa terdapat 7 CV yang cocok dengan menggunakan fuzzy match. Ini berarti, program tidak berhasil menemukan satupun hasil CV yang memiliki kata kunci python. Akibatnya, program menggunakan algoritma Levenshtein Distance, dengan batas atas 2 distance.

3. Keyword: Experience with Information Technology Service Managment
Algoritma: Boyer-Moore



Berdasarkan hasil pengujian, aplikasi berhasil menemukan 1 CV yang memiliki kata kunci yang sesuai. Menggunakan algoritma Boyer-Moore, dibutuhkan waktu 336 detik untuk pencocokan string pada seluruh file CV. Boyer Moore efisien untuk

digunakan dalam pencarian dengan pattern yang panjang. Algoritma ini menggunakan perbandingan suffix dan prefix, yang membuat perbandingan dapat “loncat” daripada membandingkan setiap karakter satu persatu. Dalam kasus terburuk, algoritma ini memiliki kompleksitas $O(n*m)$.

4. Keyword : XXXXXXXXXXXXXXXXXXXX
Algoritma : Boyer-Moore



Program tidak berhasil menemukan CV yang sesuai dengan kata kunci. Algoritma yang dipilih, Boyer-Moore, hanya mencari string yang memiliki substring yang sama persis dengan keyword. Saat tidak menemukan hasil, program akan beralih ke fuzzy matching. Di sini, program akan mencoba mencari Levenshtein Distance terpendek yang dapat dihasilkan dari string text menuju keyword. Namun, dengan keyword yang cukup panjang, mustahil untuk bisa didapatkan keyword dari substring text CV menggunakan 2 distance. Akibatnya, dengan fuzzy matching, program juga gagal menemukan hasil.

Bab 5

Kesimpulan dan Saran

a. Kesimpulan

Pada tugas besar ini, penulis telah membangun ATS (Applicant Tracking System) berbasis CV digital dengan menggunakan algoritma Boyer-Moore dan Knuth-Morris-Pratt (KMP). Aplikasi ini memudahkan perekrut untuk mencari kandidat yang sesuai untuk perusahaannya karena mendukung banyak fitur seperti.

- a. Fitur ekstraksi teks dari file PDF yang secara otomatis membaca file PDF dari path yang tersimpan di database pelamar.
- b. Fitur pencarian keyword yang memungkinkan pengguna untuk memasukkan satu atau lebih keyword yang ingin di cari dari daftar CV pelamar.
- c. Fitur pemilihan algoritma yang memungkinkan pengguna untuk menentukan algoritma pencarian (KMP dan BM).
- d. Fitur fuzzy matching yang memungkinkan pencocokan kata kunci dengan Levenshtein Distance untuk mencari kata yang mirip.
- e. Fitur perhitungan dan penyajian hasil pencarian yang memungkinkan aplikasi menampilkan CV hasil pencarian berdasarkan jumlah kemunculan kata kunci dan tingkat kemiripan.
- f. Fitur membuka file CV yang memungkinkan pengguna untuk membuka CV seorang pelamar yang mengandung pola seperti keyword.
- g. Fitur ringkasan profil yang memungkinkan pengguna untuk melihat ringkasan profil CV yang mengandung pola seperti keyword.

b. Saran

Dari tugas besar ini, tentunya program yang penulis buat belum sempurna dan masih memiliki banyak keterbatasan. Untuk itu, penulis mengharapkan kritik dan saran dari pihak-pihak yang akan mengevaluasi tugas ini. Implementasi algoritma seperti Boyer-Moore (BM) dan Knuth-Morris-Pratt serta ekstraksi dengan regex pada aplikasi ATS berbasis CV digital masih bisa ditingkatkan untuk hasil yang lebih optimal.

LAMPIRAN

Tautan *repository* GitHub : [juliusev](#)

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 <i>Knuth-Morris-Pratt (KMP)</i> dan <i>Boyer-Moore (BM)</i> dapat menemukan kata kunci dengan benar.	✓	
5	Algoritma <i>Levenshtein Distance</i> dapat mengukur kemiripan kata kunci dengan benar.	✓	
6	Aplikasi dapat menampilkan <i>summary CV applicant</i> .	✓	
7	Aplikasi dapat menampilkan <i>CV applicant</i> secara keseluruhan.	✓	
8	Membuat laporan sesuai dengan spesifikasi.	✓	
9	Membuat bonus enkripsi data profil <i>applicant</i> .		✓
10	Membuat bonus algoritma <i>Aho-Corasick</i> .		✓
11	Membuat bonus video dan diunggah pada Youtube.		✓

DAFTAR PUSTAKA

- Munir, R. (2025). *Pencocokan string (string matching) dengan algoritma brute force, KMP, Boyer-Moore*. Homepage Rinaldi Munir. Retrieved May, 2025, from [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/23-Pencocokan-string-\(2025\).pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/23-Pencocokan-string-(2025).pdf)
- Munir, R. (2025). *Pencocokan string dengan regular expression (regex)*. Homepage Rinaldi Munir. Retrieved May, 2025, from [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/24-String-Matching-dengan-Regex-\(2025\).pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/24-String-Matching-dengan-Regex-(2025).pdf)