PENERAPAN WEB SCRAPING SEBAGAI MEDIA PENCARIAN BERDASARKAN KEYWORD (STUDI KASUS : METODE PENGEMBANGAN SISTEM INFORMASI AKADEMIK)

Maya Sukanti: Diah Priyawati, S. T., M. Eng

Program Studi Teknik Informatika, Fakultas Komunikasi dan Informatika, Universitas Muhammadiyah Surakarta

Abstrak

Pencarian literatur ilmiah terkait metode pengembangan sistem informasi akademik pada Google Scholar sering menghadapi kendala efisiensi akibat proses pencarian secara manual yang akan memakan waktu. Penelitian ini bertujuan untuk menerapkan teknik web scraping sebagai media pencarian otomatis berdasarkan kata kunci (keyword) dengan studi kasus metode pengembangan sistem informasi akademik menggunakan Google Scholar sebagai sumber data. Penerapan web scraping ini digunakan untuk mempercepat pengumpulan data artikel ilmiah. Sistem dibangun menggunakan bahasa pemrograman Python dengan memanfaatkan library Requests untuk menangani permintaan HTTP dan library BeautifulSoup untuk melakukan parsing HTML. Data yang diekstraksi disimpan ke dalam format file Comma-Separated Values (CSV). Hasil penelitian menunjukkan bahwa implementasi web scraping berbasis Python ini mampu memangkas waktu pencarian literatur secara signifikan dibandingkan dengan metode manual. Dataset yang dihasilkan siap digunakan untuk analisis literatur lanjutan.

Kata Kunci : Web scraping, Python, Requests, BeautifulSoup, Google Scholar, Sistem Informasi Akademik

Abstract

Searching for scientific literature related to academic information system development methods on Google Scholar often faces efficiency constraints due to the time-consuming manual search process. This study aims to apply web scraping technique as an automatic search medium based on keyword with a case study of academic information system development methods using Google Scholar as a data source. The application of web scraping is used to accelerate the collection of scientific article data. The system is built using the Python programming language by utilizing the Requests library to handle HTTP Requests and BeautifulSoup library to parse HTML. The extracted data saved into a Comma-Separated Values (CSV) file format. The results show that the implementation of Python-based web scraping is able to reduce literature search time significantly compared to the manual method. The resulting dataset is ready to be used for further literature analysis.

Keyword : Web scraping, Python, Requests, BeautifulSoup, Google Scholar, Academic Information System**.**

1. PENDAHULUAN

Perkembangan teknologi informasi yang berkembang semakin pesat dan peredaran informasi yang semakin banyak dan kompleks membuat dunia pendidikan harus mampu mengikuti perkembangan teknologi terutama dalam bidang komputer. Perkembangan tersebut telah membawa perubahan mendasar bagi manusia untuk mengakses, mengelola dan memanfaatkan informasi. Internet menjadi infrastruktur utama dalam kehidupan modern, termasuk di sektor pendidikan. Setiap tahun, pengguna internet di Indonesia melonjak cukup signifikan. Lonjakan ini menunjukkan bahwa sistem digital di Indonesia terus berkembang pesat. Hal-hal terkait administrasi, tuntutan kebutuhan informasi, perubahan prosedur menjadi alasan mendasar pentingnya keberadaan sistem informasi di dunia pendidikan. Sistem informasi tersebut harus selalu dikembangkan dengan metode pengembangan sistem yang sesuai dengan sistem yang telah berjalan agar kinerja sistem bekerja dengan baik. Pengembangan sistem berarti melakukan perubahan sistem dengan menyusun dan menggantikan sistem lama dengan sistem baru, melakukan perbaikan sistem yang bermasalah, atau melakukan perbaikan sistem secara keseluruhan.

Seiring dengan pengembangan sistem informasi akademik, kebutuhan terhadap sistem pencarian informasi yang cepat, akurat dan efisiensi menjadi semakin kritis. Penerapan digitalisasi pada sektor pendidikan telah mengubah layanan akademik secara menyeluruh, dimana sistem informasi memungkinkan berbagai proses akademik berjalan lebih efisien, transparansi layanan dan penghematan biaya operasional serta kemudahan akses bagi seluruh sivitas akademik. Namun, perkembangan ini menimbulkan ketidak sesuaian tersendiri, dimana semakin banyak informasi yang tersedia secara digital, maka semakin sulit pula menemukan informasi yang benar-benar relevan dan spesifik sesuai kebutuhan. Mahasiswa atau peneliti yang ingin mempelajari metode pengembangan sistem informasi akademik harus menghadapi kenyataan bahwa informasi tersebut tersebar di ratusan bahkan ribuan halaman web dari berbagai institusi, situs jurnal atau repositori daring tidak saling terintegrasi.

Keterbatasan mekanisme pencarian informasi untuk menjawab kebutuhan yang spesifik dan terstruktur tersebut yang menjadi permasalahan nyata yang akan dihadapi mahasiswa maupun peneliti. Pencarian informasi menggunakan mesin pencari umum seperti Google memang menghasilkan banyak keluaran informasi. Namun, keluaran tersebut seringkali tidak terfilter, tidak terklasifikasi, dan memerlukan waktu yang

cukup lama karena dilakukan secara manual. Memasukkan istilah tanpa taktik atau belum dilakukan secara sistematis, menyebabkan mesin pencari memberikan hasil penelusuran tanpa adanya batasan. Kondisi ini diperburuk dengan fakta bahwa tidak ada sistem pencarian terpusat yang khusus dirancang untuk mengumpulkan dan mengklasifikasikan informasi terkait metode pengembangan sistem informasi akademik secara otomatis. Data yang diperoleh pada umumnya berada dalam format yang tidak terstruktur sehingga dilakukan langkah-langkah pra-pemrosesan sebelum data tersebut dimanfaatkan.

Seiring penerapan digitalisasi pada sektor pendidikan, kebutuhan mahasiswa informatika dan para peneliti terhadap referensi yang terstruktur terkait dengan metode pengembangan sistem informasi akademik juga semakin tinggi. Di lain sisi, teknologi *web scraping* telah berkembang dan memberikan solusi yang terstruktur dan efisien untuk mengatasi masalah pengumpulan data dari berbagai sumber web secara otomatis.

Sistem Informasi Akademik adalah sistem terkomputerisasi yang dilengkapi fitur-fitur yang secara khusus dirancang mampu mendukung semua kegiatan akademik yang sesuai untuk kebutuhan pengisian data tertentu. Sistem informasi akademik bertujuan untuk mempermudah penginputan data secara akurat dan efisiensi waktu yang digunakan. Pengembangan sistem informasi memiliki metodologi dengan model yang beragam mulai dari yang terstruktur hingga berbasis objek. Dengan metode pengembangan sistem inilah akan ditentukan sistem informasi akan berjalan di perangkat berbasis web, mobile, atau desktop (Triandini et al., 2019).

Web scraping atau *web crawling* adalah prosedur ekstraksi data otomatis dari situs web menggunakan perangkat lunak khusus untuk menghasilkan data yang jauh lebih menyeluruh, akurat, dan konsisten dibandingkan entri data manual (Khder, 2021). *Web scraping* tidak dapat dimasukkan dalam bidang *data mining* karena *data mining* menyiratkan upaya untuk memahami pola semantik atau tren dari sejumlah besar data yang telah diperoleh. Aplikasi *web scraping* (juga disebut *intelligent, automated, or autonomous agents*) hanya fokus pada cara memperoleh data melalui pengambilan dan ekstraksi data dengan ukuran data yang bervariasi (Josi et al.,).

Berdasarkan uraian diatas, penelitian ini bertujuan untuk membangun sebuah sistem pencarian berbasis *web scraping* yang mampu secara otomatis mengumpulkan informasi terkait metode pengembangan sistem informasi akademik dari sumber web yaitu Google Scholar. Sistem akan menerapkan mekanisme pencarian dengan kata

kunci sehingga mendapat informasi yang relevan secara cepat dan terstruktur, serta memberikan kontribusi praktis bagi mahasiswa informatika dan peneliti dalam melakukan studi literatur pada bidang akademik secara lebih terstruktur. Penelitian ini menggunakan *web scraping* dengan *HTML Parser* yang akan membantu proses penarikan data berdasarkan tag *HTML* pada kode website. Bahasa pemrograman yang digunakan adalah *Python* dan menggunakan *library Requests, library Selenium* dan *library BeautifulSoup.* Pada proses *scraping* dilakukan identifikasi tag *HTML*, setelah proses *scraping* selesai data akan disimpan dalam bentuk format file CSV, kemudian file CSV tersebut akan dianalisis lebih lanjut.

2. METODE

2.1 Jenis Penelitian

Penelitian ini merupakan penelitian terapan dengan pendekatan kualitatif-deskriptif. Metode penelitian penerapan dipilih karena berfokus pada penerapan teknologi *web scraping* berbasis *Python* sebagai solusi praktis untuk otomatisasi ekstraksi data secara efisien (Amanny Ulfah Nabiylah Ramadhanty & Ina Najiyah, 2023). Selanjutnya, pendekatan kualitatif-deskriptif digunakan untuk mengurai alur kerja arsitektur sistem pengumpul data serta mendeskripsikan karakteristik struktural dari dataset yang berhasil diekstrak (Alfiansyah et al., 2026).

2.2 Alat dan Bahan Penelitian

Penelitian ini menggunakan perangkat keras berupa laptop atau komputer dengan spesifikasi minimum prosesor Intel Core i3 generasi ke-8 dengan RAM 8 GB dan koneksi internet stabil. Adapun perangkat lunak yang digunakan antara lain *Python* sebagai bahasa pemrograman utama, Visual Studio Code sebagai editor kode (IDE), *library Requests* sebagai pengiriman permintaan HTTP ke server, *library BeautifulSoup* sebagai *parsing* dan ekstraksi elemen *HTML*, *library Selenium* sebagai manipulasi data dan ekspor ke file CSV, *library time* sebagai pengaturan *delay* antar permintaan, dan Google Scholar sebagai sumber data utama.

2.3 Implementasi *Web Scraping* pada Berbagai Bidang

Hafiz & Sudarmilah, (2023) mengimplementasikan *web scraping* pada portal berita CNBC Indonesia, CNN Indonesia, Kompas, Republika menggunakan *Python* dan *BeautifulSoup*. Penelitian ini menyimpulkan bahwa *web scraping* dapat mempercepat proses pengumpulan data secara signifikan dibandingkan pengumpulan manual dan hasil *scraping* dapat

dimanfaatkan untuk analisis sentimen dan identifikasi tren berita terkini. Mitra et al., (2017) merancang aplikasi *web scraping* berbasis *HTML DOM* untuk membangun korpus paralel teks Indonesia-Inggris dari situs web bilingual. Data korpus hasil *scraping* digunakan sebagai bahan pelatihan sistem penerjemahan mesin. Arief & Kurniawan, (2020) membangun sistem otomatis yang melakukan *scraping* harga komoditas dari berbagai sumber online, menyimpannya ke database, menampilkannya dengan dashboard real-time, dan menambahkan fitur prediksi harga kemudian diuji fungsionalitasnya dan mendapatkan skor cukup baik.

Kusumo, (2022) merancang sistem *web scraping* deskripsi dari Tokopedia, Bukalapak, Lazada, Amazon, dan Alibaba menggunakan *Python* dan PHP dalam *framework* Laravel. Teknik ini terbukti mampu mengambil konten deskripsi produk secara cepat dan menyimpannya ke database. Ramadan & Handaga, (2019) mengembang kan sebuah aplikasi mobile berbasis Android yang memanfaatkan teknik *web scraping* untuk mengekstrak data akademik mahasiswa Universitas Muhammadiyah Surakarta dari laman Forlap Dikti. Pengujian dilakukan menggunakan metode *black box testing* dan kuesioner 30 responden yang menyatakan aplikasi layak digunakan sebagai alternatif akses data Forlap Dikti melalui perangkat smartphone.

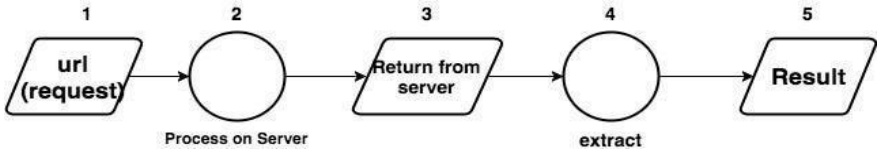
2.4 Objek dan Sumber Data Penelitian

Sumber data penelitian adalah basis data pencarian Google Scholar, yang memuat metadata artikel ilmiah berupa judul, penulis, tahun terbit, nama publikasi, jumlah sitasi, tautan (link), dan cuplikan abstrak. Kata kunci pencarian yang digunakan adalah “metode pengembangan sistem informasi akademik”, dengan cakupan pengambilan data sebanyak 5 halaman hasil pencarian (maksimal 10 artikel per halaman, sehingga hasil pencarian maksimal 50 artikel per metode sebelum proses deduplikasi).

2.5 Perancangan Sistem

Sistem dirancang dengan tiga komponen utama yaitu komponen *input*, komponen proses, dan komponen *output*. Komponen *input* yaitu pengguna memasukkan satu atau lebih kata kunci melalui antarmuka berbasis terminal. Komponen proses yaitu proses utama sistem yang dilakukan oleh mesin *scraping* yang bekerja secara otomatis untuk mengekstrak data relevan (Rizquina & Ratnasari, 2023). Komponen *output* yaitu hasil ekstraksi data dari proses *scraping*, pada penelitian ini hasil ekstraksi disimpan dalam format file CSV yang memuat kolom-kolom data yang terstruktur. Format file CSV dipilih karena mudah dibaca, ringan, dan kompatibel dengan berbagai perangkat lunak

analisis data(Muhammad Adhit Dwi Yuda, 2025). Alur perancangan sistem atau alur proses *scraping* dapat dilihat pada Gambar 1.



Gambar 1 Alur Perancangan Sistem

2.6 Tahapan Penelitian

Penelitian ini dimulai dengan penentuan kata kunci pencarian, inisialisasi komponen scraper (sesi HTTP untuk *Requests* dan *instance* WebDriver untuk *Selenium*), kemudian kedua metode *scraping (Requests+BS4* dan *Selenium+BS4)* dijalankan untuk mengumpulkan data mentah berupa halaman HTML. Data tersebut selanjutnya di-*parsing* dan dibersihkan menggunakan pola *regular expression* (*regex*), digabungkan, dideduplikasi, disimpan ke file format CSV, dan terakhir dievalusi tingkat akurasinya sebelum dinyatakan selesai. Setelah penulisan kode selesai, akan dilakukan pengujian terhadap sistem. Apabila pada tahap pengujian sistem belum berjalan dengan baik maka akan dilakukan perbaikan, sehingga siklus ini bersifat iteratif hingga sistem menghasilkan keluaran yang valid (Satriajati et al., 2020).

2.6.1 Metode Pengumpulan Data

Pengumpulan data primer dilakukan dengan teknik *web scraping*, yaitu proses pengambilan data secara otomatis dari halaman web dengan mengurai (*parsing*) struktur HTML. Pada penelitian ini, proses *scraping* dirancang menggunakan dua metode yang berbeda yaitu *Requests* dan *Selenium*, dimana keduanya dikombinasikan dengan *BeautifulSoup*. Pemilihan dua metode *scraping* yang berbeda didasarkan pada karakteristik dan keterbatasan masing-masing metode ketika berhadapan dengan mekanisme proteksi Google Scholar terhadap aktivitas (bot).

Requests+BeautifulSoup dipilih karena efisiensi sumber daya dan kecepatan eksekusi, metode ini hanya mengirimkan permintaan HTTP dan mengurai respons *HTML* tanpa perlu me-*render* JavaScript maupun membuka *instance* browser sehingga proses berjalan lebih ringan dan cepat. Kelemahannya, Google Scholar dapat mendeteksi pola permintaan berulang dari sesi HTTP sederhana dan meresponnya dengan kode status 429 (*Too Many Requests*) atau halaman CAPTCHA. *Selenium+BeautifulSoup* dipilih sebagai metode cadangan sekaligus pelengkap karena mampu menjalankan browser sungguhan (*headless* Chrome) yang me-*render* halaman secara utuh termasuk JavaScript, serta menyamarkan indikator otomatisasi (misalnya navigator.webdriver) melalui konfigurasi

opsi khusus, metode ini membuat permintaan terlihat lebih menyerupai perilaku pengguna (manusia) sehingga meminimalkan kemungkinan diblokir. Kelemahannya, proses ini lebih berat secara komputasi dan lebih lambat karena harus menunggu proses *rendering* serta pemuatan elemen halaman. Jika salah satu metode terhambat misalnya *Requests* terkena rate-limit, metode lainnya yaitu *Selenium* tetap dapat melanjutkan pengumpulan data pada sesi yang sama sehingga risiko kegagalan total dalam pengumpulan data dapat diminimalkan. Kedua metode tidak diposisikan sebagai dua kelompok eksperimen yang dibandingkan secara terpisah untuk memilih yang lebih andal, melainkan dijalankan secara berurutan dan saling melengkapi dalam satu sesi pengumpulan data yang sama.

```
FUNGSI scrape_dengan_requests(query, max_pages)
semua_hasil ← []
sesi ← buat sesi HTTP baru
terapkan header User-Agent acak pada sesi
UNTUK halaman DARI 0 HINGGA max_pages – 1:
start ← halaman × 10
params ← {q: query, start: start, hl: 'id'}
COBA:
        resp ← sesi.GET(BASE_URL, params, timeout=15s)
        JIKA resp.status == 429 → tunggu 30 detik,lanjut
        JIKA resp.status != 200 → catat error, lanjut
        soup ← parse HTML dari resp.text
        artikel ← parse_artikel(soup, 'requests+BS4')
        tambahkan artikel ke semua_hasil
        TANGKAP error jaringan → catat pesan
delay_acak() // jeda 2-4 detik
KEMBALIKAN semua hasil
```

Gambar 2 *Pseudocode Scraping* dengan *Requests*

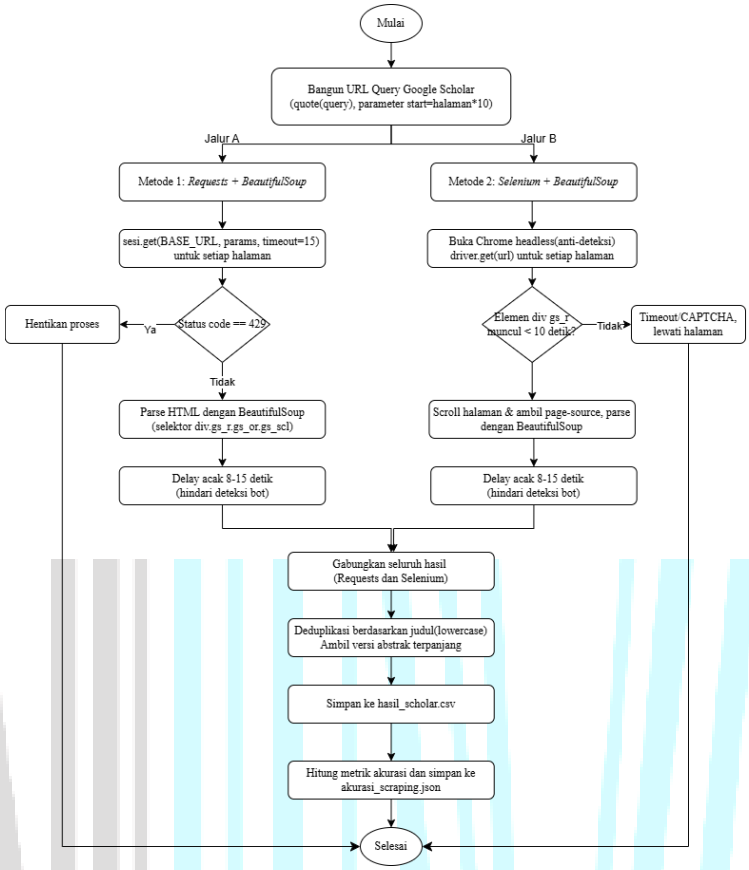
Setiap artikel hasil *scraping* akan diberi label pada *field* metode *scraping* pada file hasil_scholar.csv (bernilai "*Requests*+BS4", "*Selenium*+BS4", atau gabungan keduanya apabila artikel yang sama berhasil diperoleh oleh kedua metode), sehingga kontribusi masing-masing metode tetap dapat ditelusuri dan dianalisis secara deskriptif tanpa perlu menjalankan keduanya sebagai eksperimen terpisah. *Pseudocode scraping* dengan metode 1 (*Requests* + *BeautifulSoup*) ditampilkan pada Gambar 2, dan *Pseudocode scraping* dengan metode 2 (*Selenium* + *BeautifulSoup*) ditampilkan pada Gambar 3.

```
FUNGSIbuat_driver():
opsi ← konfigurasi Chrome headless dengan anti-deteksi
pasang ChromeDriver via ChromeDriverManager
hapus properti navigator.webdriver via JavaScript
KEMBALIKAN driver
FUNGSI scrape_dengan_selenium(query, max_pages):
 COBA membuka browser:
 driver ← buat_driver()
 JIKA gagal → KEMBALIKAN []
 COBA:
 UNTUK halaman DARI 0 HINGGA max_pages - 1:
 url ← BASE_URL + '?q=' + query + '&start=' +(halaman×10)
 driver.buka(url)
 TUNGGU elemen div.gs_r muncul (maks 10 detik)
 JIKA timeout → lanjut ke halaman berikutnya
 scroll halaman ke bawah (lazy-load trigger) TUNGGU 1 detik
 soup ← parse HTML dari driver.page_source
 artikel ← parse_artikel(soup, 'selenium+BS4')
 tambahkan artikel ke semua_hasil delay_acak()
 AKHIRNYA:
 driver.quit() // selalu tutup browser
 KEMBALIKAN semua_hasil
```

Gambar 3 *Pseudocode Scraping* dengan *Selenium*

2.6.2 Implementasi *Scraping*

Implementasi sistem atau proses *web scraping* dilakukan menggunakan aplikasi Visual Studio Code sebagai editor kode atau *Intregated Development Environment* (IDE). Sebelum proses *scraping* dilakukan identifikasi kelas tag *HTML*, yaitu tag yang mengapit informasi yang ingin diekstraksi untuk kemudian dibuatkan template *scraping* (Mitchell, 2018). Identifikasi tag *HTML* digunakan untuk memetakan struktur dokumen *HTML* target untuk memisahkan konten inti dari elemen gangguan (teks pengganggu serta iklan, sidebar, dan footer). Pembangunan URL utama adalah tahap inisiasi *query*. Sebagian besar mesin pencari akademik menggunakan metode *HTTP GET*, dimana parameter pencarian disematkan langsung pada URL (Falah & Suryawan, 2021). Alur teknis implementasi kedua metode *scraping*, dimulai dari pembangunan URL permintaan hingga tahap evaluasi akurasi akhir, ditampilkan pada Gambar 4.



Gambar 4 Alur Implementasi *Scraping*

Berdasarkan Gambar 4, kedua jalur (Jalur A untuk *Requests* dan Jalur B untuk *Selenium*) memiliki pola kerja yang sejajar yaitu membangun URL pencarian dengan parameter halaman (start = halaman x 10), mengambil respons halaman, memeriksa kondisi kegagalan spesifik pada masing-masing metode (kode status 429 pada Jalur A dan batas waktu tunggu elemen pada Jalur B), mem*parsing* hasil yang diperoleh, lalu memberi jeda acak sebelum berpindah ke halaman berikutnya. Setelah seluruh halaman pada kedua jalur selesai diproses, hasilnya disatukan pada tahap penggabungan, dilanjutkan dengan deduplikasi, sebelum tahap akhir dilakukan pengujian sistem untuk meminimalkan terjadinya kesalahan ekstraksi data, kemudian data disimpan dalam bentuk sesuai kebutuhan yaitu format file CSV untuk memudahkan proses pra-pemrosesan dan analisis data pada tahap berikutnya (Glez-Peña et al., 2013), dan tahap akhir yaitu penghitungan skor akurasi. Jalur A atau metode pertama menggunakan *library Requests* untuk penarikan dokumen berbasis halaman statis secara cepat (Muhammad Adhit Dwi Yuda, 2025). Jalur B atau metode kedua menggunakan *library Selenium* untuk menangani elemen antarmuka halaman dinamis yang di*render* menggunakan JavaScript (Wijaya & Budiaji, 2025).

Library Requests digunakan untuk menangani *header HTTP*, *session management* dan respon server. Sementara *library Selenium* digunakan untuk mengeksekusi konten yang dimuat secara dinamis oleh JavaScript. *Library BeautifulSoup* digunakan untuk melakukan proses *parsing* terhadap konten *HTML* yang diterima dari server, *BeautifulSoup* mengekstraksi elemen-elemen yang relevan menggunakan metode pencarian berdasarkan nama tag, atribut, maupun kelas *Cascading Style Sheets (CSS)* (Muhammad Adhit Dwi Yuda, 2025). *Library time* dan *random* digunakan untuk memberikan jeda waktu (*delay*) yang bersifat acak antar setiap permintaan untuk mengantisipasi sistem pertahanan server sehingga sistem dapat menghindari pemblokiran oleh mekanisme anti-bot Google Scholar (Rifka Alkhilyatul Ma'rifat, I Made Suraharta, 2024).

2.6.3 Teknik *Parsing* dan Ekstraksi Data

Setelah halaman HTML berhasil diunduh, kemudian melakukan *HTML parsing* dengan *BeautifulSoup* untuk mentransformasikan teks mentah *HTML* menjadi objek terstruktur, sehingga informasi yang ditargetkan dapat disaring sesuai template yang telah dibuat (Richardson, 2016). Setiap halaman HTML yang berhasil diambil (baik dengan metode *Requests* maupun dengan metode *Selenium*), di-*parsing* menggunakan fungsi parse_artikel() yang sama agar struktur data yang dihasilkan konsisten. Fungsi ini menargetkan elemen kartu artikel dengan selektor CSS div.gs_r.gs_or.gs_scl, kemudian mengekstraksi tujuh *field* dari setiap kartu yaitu judul dan link (dari elemen h3.gs_rt a), metadata penulis/tahun/publikasi (dari elemen div.gs_a), abstrak (dari elemen div.gs_rs), serta jumlah sitasi (dari tautan pada div.gs_fl yang mengandung kata kunci "dirujuk", "dikutip", atau "cited by"). Karena teks yang diambil dari halaman web sering mengandung spasi berlebih, potongan kalimat yang tidak terstuktur, maupun format angka yang bercampur dengan teks (misalnya "Dikutip oleh 1.282"), maka proses ekstraksi *field-field* tersebut dibantu oleh sejumlah pola*regular expression* (*regex*). Rincian pola *regex* yang digunakan beserta penjelasan fungsinya disajikan pada Tabel 1.

Tabel 1 Penjelasan Kode *Regex*

No	Digunakan pada Fungsi	Pola *Regex*	Penjelasan
1	Bersihkan_teks()	\s+	Mencocokkan satu atau lebih karakter spasi/whitespace secara beruntun. Digunakan bersama re.sub() untuk mengganti seluruh whitespace berlebih menjadi satu spasi tunggal, sehingga teks judul, penulis, dan abstrak menjadi rapi sebelum disimpan.
2	Ekstrak_tahun	\b(19|20)\d{2}\b	Mencari pola 4 digit angka yang diawali "19" atau "20" dan dibatasi oleh word boundary (\b) agar tidak salah menangkap potongan angka dari string

No	Digunakan pada Fungsi	Pola *Regex*	Penjelasan
			lain. Pola ini efektif menangkap tahun terbitan pada rentang 1900-2099 dari teks metadata artikel (div.gs_a).
3	Ekstrak_sitasi()	(?:Dikutip|Cited by|Dirujuk)[^\d]{0,15}([\d.,]+)	Mendeteksi teks penanda jumlah sitasi dalam tiga variasi bahasa/ istilah yang dipakai Google Scholar ("Dikutip oleh", "Cited by", "Dirujuk". Grup non-capturing (?...) memilih salah satu kata kunci tanpa ikut diambil sebagai hasil, [^\d]{0,15} melompati maksimal 15 karakter non-digit sebelum akhirnya grup tangkap ([\d.,]+) mengambil deretan angka beserta tanda pemisah ribuan
4	Ekstrak_sitasi()	[.,]	Menghapus tanda titik atau koma pemisah ribuan dari angka sitasi yang telah ditangkap (contoh "1.282" menjadi "1282") menggunakan re.sub() agar nilai sitasi tersimpan sebagai angka murni yang dapat diproses secara numerik.
5	Validasi_format()	^(19|20)\d{2}$	Digunakan untuk memvalidasi bahwa nilai *field* tahun benar-benar berformat 4 digit tahun yang valid (diawali ^ dan diakhiri $ agar seluruh string harus cocok, bukan sebagian). *Field* yang lolos validasi dihitung sebagai persentase tahun_valid_ persen pada laporan akurasi.

Penggunaan *regex* berperan penting dalam menjaga kualitas data karena struktur halaman Google Scholar tidak sepenuhnya konsisten seperti terdapat variasi urutan penulis-tahun-publikasi, variasi istilah jumlah sitasi dalam bahasa Indonesia maupun Inggris, serta karakter whitespace yang tidak sama akibat markup HTML. Dengan pola-pola *regex* tersebut, *field* tahun dan sitasi dapat diekstraksi secara presisi dari teks bebas (free text), sementara fungsi bersihkan_teks() memastikan seluruh *field* bebas dari spasi ganda maupun karakter baris baru yang mengganggu keterbacaan maupun proses analisis lanjutan.

Pola *regex* \b(19|20)\d{2}\b pada Tabel 1 mendefinisikan rentang format tahun yang dianggap valid untuk dikenali yaitu 1900-2099 dan bukan merupakan prediksi atau batasan terhadap rentang tahun terbitan artikel yang akan diperoleh. Pola ini bersifat inklusif dan hanya berfungsi sebagai saringan format (menolak string yang bukan 4 digit tahun) sehingga ia tidak menolak maupun memaksa data untuk berada pada sub-rentang tertentu di dalam batas tersebut. Pada hasil *scraping* rentang tahun terbitan yang ditemukan yaitu sekitar tahun 2010-2026, semata-mata merupakan cerminan dari karakteristik data yang tersedia di Google Scholar untuk kata kunci "metode pengembangan sistem informasi akademik", mesin pencari ini menampilkan hasil berdasarkan tingkat relevansi dan jumlah sitasi (bukan urutan kronologis), sedangkan topik pengembangan sistem informasi akademik sendiri baru berkembang pesat sebagai bahan penelitian sekitar tahun 2010-an. Dengan demikian, sempitnya rentang tahun pada data hasil (2010-2016) dibandingkan dengan rentang yang secara format dapat dikenali oleh *regex* (1900-2099) bukan merupakan indikasi kesalahan maupun inkonsistensi pada kode program, melainkan wajar terjadi karena *regex* hanya

memvalidasi format kemunculan tahun pada teks, sementara sebaran nilainya sepenuhnya ditentukan oleh data artikel yang tersedia dan relevan di sumber data itu sendiri. Selain topik pengembangan sistem informasi akademik yang baru berkembang pesat sebagai bahan penelitian sekitar tahun 2010-an, rentang tahun pada data hasil hanya berada di rentang 2010 sampai 2026 merupakan karena pada penelitian ini proses *scraping* hanya dibatasi dalam 5 halaman dimana setiap halaman maksimal hanya 10 aartikel, jika ditotal maksimal data yang diperoleh yaitu 50 artikel.

2.6.4 Proses Penggabungan dan Deduplikasi Data

```
FUNGSI gabungkan_dan_deduplikasi(hasil_req,hasil_sel):
 gabungan ← {} // kamus: kunci = judul lowercase
 duplikat ← 0
 UNTUK SETIAP artikel DALAM (hasil_req + hasil_sel):
 kunci ← artikel['judul'].lowercase()
 JIKA kunci TIDAK ADA di gabungan:
 gabungan[kunci] ← artikel
 JIKA TIDAK (duplikat):
 duplikat ← duplikat + 1
JIKA len(artikel['abstrak']) >
  len(gabungan[kunci]['abstrak']):
  metode_lama gabungan[kunci]['metode_scraping']
  gabungan[kunci] ← artikel
  gabungan[kunci]['metode_scraping']← metode_lama + ' + ' + artikel ['metode_scraping']
  cetak jumlah duplikat yang dihapus
  daftar ← list(gabungan.values())
  beri nomor urut (no) pada setiap artikel KEMBALIKAN daftar
```

Gambar 5 *Pseudocode* Deduplikasi

Setelah data dari kedua metode terkumpul, selanjutkan adalah menggabungkan seluruh hasil (hasil_req dan hasil_sel) ke dalam satu struktur data melalui fungsi gabungkan_dan_deduplikasi(). Proses ini menggunakan judul artikel dalam format huruf kecil (*lowercase*) sebagai kunci unik pembanding. Apabila ditemukan dua entri dengan judul identik maka sistem akan mempertahankan entri dengan abstrak yang lebih panjang, karena abstrak yang lebih panjang dianggap lebih informatif. *Pseudocode* gabungkan_dan_deduplikasi data dapat dilihat pada gambar 5.

2.6.5 Pengujian Sistem

Setelah penulisan kode *web scraping* selesai dilakukan pengujian terhadap sistem. Pengujian sistem pada penelitian ini menggunakan pengujian fungsional (*Black-Box testing*).

Pengujian fungsional bertujuan untuk memverifikasi bahwa setiap fungsi bekerja sesuai dengan spesifikasi yang telah dirancang yang meliputi pengiriman permintaan berhasil, proses penguraian *HTML* menghasilkan data yang benar, ekstraksi data lengkap, dan file CSV tersimpan dengan format yang tepat (Falah & Suryawan, 2021).

2.6.6 Teknik Evaluasi Akurasi Data

Untuk mengukur kualitas data hasil *scraping* yang tersimpan dalam file CSV, dilakukan pengujian kualitas informasi yang dilakukan secara matematis menggunakan empat indikator evaluasi utama yaitu kelengkapan *field*, relevansi artikel, validasi format data dan skor akurasi keseluruhan. Kelengkapan *field* (kelengkapan_*field*_%) yaitu persentase keterisian setiap *field* penting (judul, penulis, tahun, publikasi, sitasi, link, abstrak) terhadap keseluruhan data. Relevansi (relevansi) yaitu skor kesesuaian judul dan abstrak artikel terhadap kata kunci pencarian, dengan bobot kemunculan kata kunci pada judul (bobot 2) lebih tinggi dibandingkan pada abstrak (bobot 1), kemudian dinormalisasi ke skala 0-100%. Validitas format (validasi_format_%) yaitu pemeriksaan format tahun harus 4 digit (1900-2099), link harus diawali "http atau https", dan sitasi harus berupa angka. Duplikat sisa (duplikat_sisa) yaitu jumlah judul yang masih terduplikasi setelah proses deduplikasi, sebagai indikator kegagalan penyatuan data. Keempat indikator tersebut kemudian digabungkan menjadi satu Skor Akurasi Total menggunakan formula berikut:

Skor Akurasi = (0,40 x rata-rata kelengkapan field) + (0,35 x persentase artikel relevan) + (0,20 x rata-rata validitas format) – (0,05 x penalti duplikat)

Bobot terbesar (40%) diberikan pada kelengkapan *field* karena kelengkapan data merupakan prasyarat dasar agar data dapat dianalisis lebih lanjut, relevansi diberi bobot 35% karena berkaitan langsung dengan tujuan pencarian, validitas format diberi bobot 20% sebagai indikator kebersihan data dan penalti duplikat (maksimal 5%) berfungsi sebagai pengurang nilai apabila proses deduplikasi tidak berjalan sempurna. Seluruh hasil evaluasi disimpan ke dalam file akurasi_*scraping*.json.

2.7 Legalitas *Web Scraping*

Wijayanti & Kharisma, (2024) menyampaikan penggunaan *web scraping* dalam ChatGPT merupakan tindakan yang berpotensi melanggar hak cipta karena telah mengurangi hak ekonomi dari pemilik atau pengelola *website* dengan tidak adanya perjanjian lisensi dari pemilik atau pemegang hak cipta. Brown et al., (2025) menyajikan tinjauan menyeluruh mengenai lingkungan regulasi yang berdampak pada kapan dan bagaimana para peneliti dapat mengakses, mengumpulkan, menyimpan, dan berbagi data melalui

scraping, serta memberikan rekomendasi bagi peneliti untuk melakukan *scraping* secara sah, ilmiah dan etis. Seluruh area hukum ini terus berkembang seiring pertumbuhan model bisnis berbasis data. Perusahaan media sosial dan operator situs web mengandalkan perjanjian dengan pengguna untuk melindungi hak hukum mereka, termasuk mengatur dan/atau melarang *scraping* serta bentuk pengumpulan data otomatis lainnya. Logos et al., (2023) membangun kerangka kerja etis dan legal penggunaan *web scraping* dalam penelitian keamanan siber berdasarkan tinjauan sistematis hukum Australia. Menetapkan panduan empat dimensi: Data Availability Act 2022, Privacy Act 1988, Copyright Act 1968, dan kode pidana federal dan negara bagian.

3. HASIL DAN PEMBAHASAN

3.1 Gambaran Umum Hasil Pengumpulan Data

Bab ini menyajikan hasil implementasi sistem *web scraping* dan evaluasi akurasi data hasil *scraping* yang berhasil dikembangkan menggunakan bahasa pemrograman *Python* dan Visual Studio Code sebagai penulisan kode program. Sistem dirancang untuk mengambil data artikel akademik dari Google Scholar secara otomatis berdasarkan kata kunci yang berkaitan dengan topik metode pengembangan sistem informasi akademik. Proses *scraping* yang dijalankan berhasil mengumpulkan data artikel ilmiah dari Google Scholar pada cakupan 5 halaman hasil pencarian. Setelah melalui tahap penggabungan hasil kedua metode *scraping* dan proses deduplikasi berdasarkan judul, diperoleh 44 artikel unik tanpa sisa duplikat. Gambaran umum karakteristik dataset akhir disajikan pada Tabel 2.

Tabel 2 Gambaran Umum Dataset Hasil *Scraping*

Karakteristik Dataset	Nilai
Total Artikel sebelum deduplikasi (*Requests* + *Selenium*)	44 entri
Duplikat yang disatukan pada tahap deduplikasi	0 entri
Total artikel unik pada dataset akhir	44 artikel
Rentang tahun publikasi	2012 – 2025
Jumlah sitasi tertinggi	757 sitasi
Jumlah sitasi terendah	8 sitasi
Jumlah publikasi/ sumber unik	37 sumber berbeda

Tidak ditemukannya duplikat sisa (0 entri) menunjukkan bahwa mekanisme deduplikasi berbasis pencocokkan judul (lowercase) pada fungsi gabungkan_dan_deduplikasi() bekerja

secara efektif untuk kasus uji ini.Cuplikan hasil penyimpanan data ke hasil_scholar.csv akan disajikan pada Tabel 3.

Tabel 3 Cuplikan Hasil Scraping

No	Judul	Penulis	Tahun	Publikasi	Sitasi	Link	Abstrak	Metode Scraping
1	Literature review: analisis metode perancangan sistem informasi akademik berbasis web	SF Arief, Y Sugiarti	2022	ejournal.fikom-unasman.ac.id	312	https://ejournal.fikom-unasman.ac.id/index.php/jikom/article/view/229	…Academic information system adalah suatu sistem yang … Sistem informasi akademik pengembangan sistem informasi yang sudah … Pengembangan sisteM informassi tidak terlepas	requests+BS4
2	Pengembangan Sistem Informasi Akademik Berbasis Website Menggunakan Metode Rapid Application Development (RAD)	L Santoso, J Amanullah	2022	journal.stekom.ac.id	89	https://journal.stekom.ac.id/index.php/elkom/article/view/943	… rapid application development(RAD), model pengembangan … dengan adanya pengembangan sistem informasi ini dapat … Meninjau pentingnya sistem informasi akademik berbasis …	requests+BS4
3	Pengembangan sistem informasi akademik berbasis web menggunakan metode Prototype pada Sekolah Menengah Pertama (SMP) Advent Kotamobagu	SD Pohan, SA Widiana, E Ketaren…	2024	ejournal.stmik-time.ac.id	23	https://ejournal.stmik-time.ac.id/index.php/jurnalTIMES/article/view/745	… ini adalah hasil pengembangan Sistem informasi Akademik pada … metode prototype mampu menyediakan informasi akademik yang ada di SMP Advent Kotamobagu seperti informasi …	requests+BS4

3.2 Kontribusi Masing-Masing Metode *Scraping*

Data pada Tabel 4 menunjukkan temuan penting yaitu seluruh 44 artikel pada dataset akhir eksekusi ini berlabel *Requests*+BS4, yang berarti pada sesi pengujian tersebut, metode 1 (*Requests*+BS4) berhasil memperoleh seluruh data, sedangkan metode 2 (*Selenium*+BS4) tidak menyumbangkan artikel baru maupun artikel duplikat yang tergabung. Hal ini disebabkan oleh beberapa kemungkinan kondisi antara lain yang pertama Google Scholar tidak menerapkan pemblokiran (rate-limit maupun CAPTCHA) terhadap sesi *Requests* pada saat pengujian dijalankan, sehingga seluruh 5 halaman berhasil diambil oleh Metode 1 tanpa hambatan dan proses *scraping* selesai sebelum metode 2 sempat menemukan artikel yang belum tercakup, yang kedua *Selenium* berpotensi mengalami kegagalan teknis pada lingkungan eksekusi (misalnya driver Chrome *headless* gagal diinisialisasi, batas waktu tunggu elemen div.gs_r terlampaui, atau koneksi jaringan pada saat proses *render*ing

halaman tidak stabil), sehingga tidak ada artikel yang berhasil di- *parsing* dari jalur ini, yang ketiga yaitu kedua metode mengakses query dan rentang halaman yang identik, sehingga apabila metode 2 berhasil berjalan namun tidak menghasilkan artikel di luar yang telah diperoleh metode 1, maka secara alami tidak akan ada label gabungan (*Requests*+BS4 dan *Selenium*+BS4) yang terbentuk.

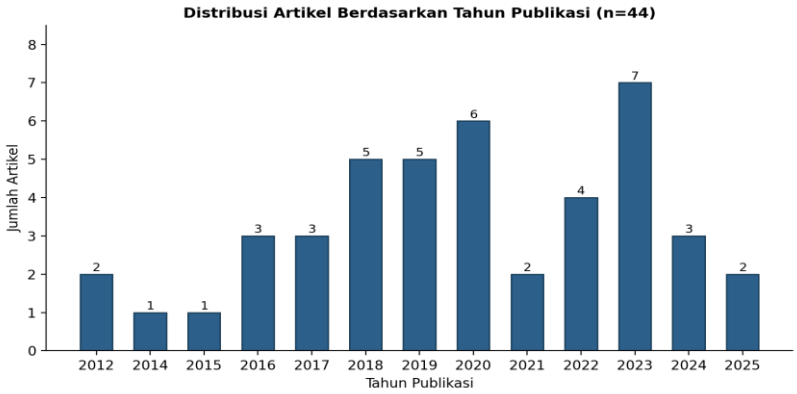
Temuan ini tidak mengurangi validitas rancangan dual-method, karena tujuan utama kombinasi kedua metode adalah sebagai mekanisme cadangan terhadap risiko pemblokiran, bukan untuk memastikan kedua metode selalu berkontribusi secara merata pada setiap sesi eksekusi. Pada eksekusi ini, keberhasilan penuh metode 1 dalam memperoleh seluruh data yang ditargetkan justru membuktikan bahwa desain sistem tetap dapat mencapai target pengumpulan data secara utuh sekalipun salah satu metode (Selenium) tidak memberikan kontribusi tambahan. Hasil penelusuran label tersebut pada dataset disajikan pada Tabel 4.

Tabel 4 Distribusi Kontribusi Metode *Scraping* terhadap Dataset Akhir

Label Metode pada Dataset	Jumlah Artikel	Persentase
Requests+BS4(hanya metode 1)	44	100%
Selenium+BS4(hanya metode 2)	0	0%
Requests+BS4 dan *Selenium*+BS4 (ditemukan kedua metode)	0	0%

3.3 Distribusi Tahun Publikasi Artikel

Distribusi artikel hasil *scraping* dengan total akhir 44 artikel berdasarkan tahun publikasi ditampilkan pada Gambar 6. Berdasarkan Gambar 6, artikel yang diperoleh tersebar pada rentang tahun antara 2012 hingga 2025 dengan konsentrasi tertinggi pada tahun 2023 sebanyak 7 artikel, diikuti tahun 2020 sebanyak 6 artikel serta tahun 2018 dan 2019 masing-masing 5 artikel. Pola ini konsisten dengan penjelasan pada Bab 2 bahwa topik pengembangan sistem informasi akademik berbasis web mulai banyak diteliti sejak pertengahan tahun 2010-an dan terus meningkat volumenya hingga beberapa tahun terakhir, sementara Google Scholar menampilkan artikel berdasarkan relevansi dan jumlah sitasi sehingga artikel-artikel populer pada rentang tahun tersebut lebih sering muncul pada halaman-halaman awal hasil pencarian.

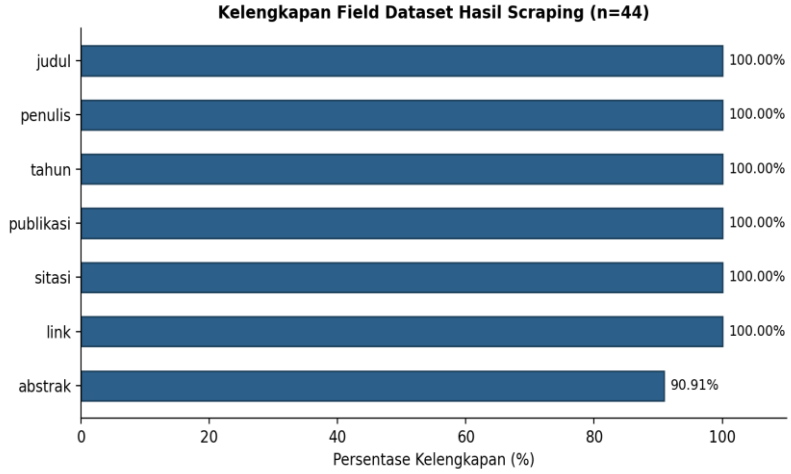


Gambar 6 Distribusi Artikel Berdasarkan Tahun Publikasi

3.4 Hasil Evaluasi Akurasi Data

3.4.1 Kelengkapan *Field*

Enam dari tujuh *field* (judul, penulis, tahun, publikasi, sitasi, dan link) memiliki kelengkapan sebesar 100%, yang berarti 44 artikel memiliki nilai terisi pada *field-field* tersebut. Satu-satunya *field* yang tidak mencapai kelengkapan penuh adalah abstrak dengan persentase 90,91% (terisi pada 40 dari 44 artikel). Empat artikel yang tidak memiliki abstrak kemungkinan disebabkan oleh elemen div.gs_rs yang memang tidak ditampilkan oleh Google Scholar untuk artikel tertentu, misalnya karena keterbatasan indeksasi pada sumber aslinya atau format halaman hasil publikasi yang berbeda dari pola umum. Persentase kelengkapan *field* divisualisasikan pada Gambar 7.



Gambar 7 Visualisasi Kelengkapan *Field*

3.4.2 Relevansi Artikel

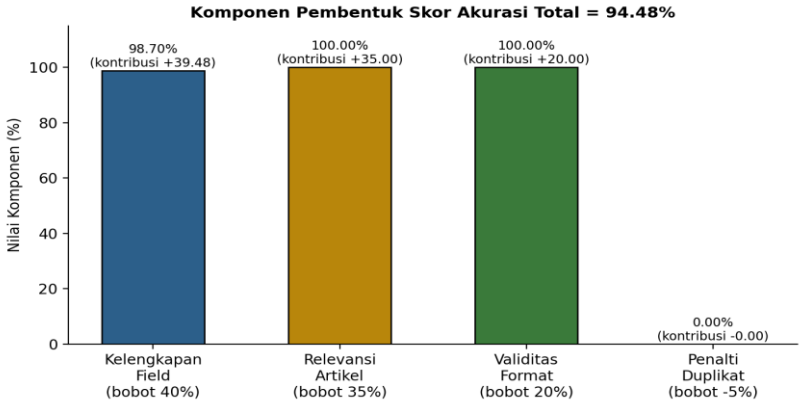
Berdasarkan perhitungan pada fungsi hitung_relevansi(), diperoleh rata-rata skor relevansi sebesar 83,18% dengan seluruh hasil *scraping* (44 artikel) dikategorikan relevan terhadap kata kunci pencarian. Hasil ini mengindikasikan bahwa kata kunci pencarian yang digunakan cukup spesifik sehingga mampu menyaring artikel-artikel yang benar-benar berkaitan

dengan pengembangan sistem informasi akademik tanpa menghasilkan artikel yang menyimpang dari topik penelitian.

3.4.3 Validitas Format

Ketiga aspek validitas format yaitu tahun, link dan sitasi seluruhnya mencapai 100% valid. Hal ini menunjukkan bahwa pola *regex* (\b(19|20)\d{2}\b yang digunakan pada tahap ekstraksi untuk tahun dan pemeriksaan awalan "http" atau "https" untuk link, serta proses pembersihan angka sitasi berhasil menghasilkan data yang seluruhnya sesuai format yang diharapkan tanpa satu pun entri yang cacat format.

3.4.4 Skor Akurasi Total



Gambar 8 Komponen Pembentuk Skor Akurasi Total

Dengan mensubstitusikan rata-rata kelengapan *field* (98,70%), persentase relevansi artikel (100%), rata-rata validitas format (100%), dan penalti duplikat (0%) ke dalam formula berbobot pada Bab 2, diperoleh perhitungan sebagai berikut (0,40 x 98,70%)+(0,35 x 100%)+(0,20 x 100%)-(0,05 x 0%) = 39,48% + 35,00% + 20,00% - 0% = 94,48%. Nilai ini termasuk dalam kategori sangat tinggi dan menunjukkan bahwa sistem *scraping* yang dibangun mampu menghasilkan data yang lengkap, relevan, valid secara format, dan bebas dari duplikasi.

3.5 Pengujian Fungsional (*Black-box Testing*)

Pengujian fungsional atau *Black-Box Testing* berfokus pada fungsi *input* dan *output* sistem tanpa melihat struktur internal kode. Hasil pengujian fungsional akan disajikan pada Tabel 5.

Tabel 5 Pengujian Fungsional

Nama Fungsi	Langkah Pengujian	Hasil yang Diharapkan	Kesimpulan
Konektivitas dan permintaan HTTP	Masukkan URL target	Sistem menerima status kode HTTP 200(OK)	Sesuai

Nama Fungsi	Langkah Pengujian	Hasil yang Diharapkan	Kesimpulan
Parsing dengan *BeautifulSoup*	Mengurai tag *HTML* untuk kemudian diekstrak	Tag *HTML* berhasil disimpan dalam variabel	Sesuai
Proses deduplikasi	Menggunakan fungsi gabungkan_dan_de duplikasi	Hasil *scraping* tidak memiliki 2 atau lebih jurnal dengan judul yang sama	Sesuai
Format tahun terbit	Menggunakan fungsi validasi_format	Tahun harus empat digit angka	Sesuai
Penyimpanan hasil *scraping*	Menggunakan fungsi simpan_ke_csv	File csv berhasil dibuat, memiliki *header* yang sesuai dan datanya cocok	Sesuai

3.6 Pembahasan

Berdasarkan keseluruhan hasil dapat dinyatakan bahwa rancangan sistem *scraping dual-method* yang dibangun pada penelitian ini berhasil mencapai tujuannya yaitu mengumpulkan data artikel ilmiah dari Google Scholar secara otomatis dengan tingkat akurasi tinggi yaitu 94,48%. Tingginya skor ini terutama didukung oleh dua faktor utama yaitu pertama ketepatan pola *regex* dalam mengekstraksi dan memvalidasi *field* tahun, sitasi, dan link secara presisi sehingga tidak ditemukan cacat format sama sekali, kemudian yang kedua yaitu spesifikasi kata kunci pencarian yang terfokus sehingga seluruh artikel yang diperoleh dinyatakan relevan dengan topik penelitian.

Efektivitas kombinasi dua metode *scraping* (*Requests* dan *Selenium*) sangat bergantung pada kondisi lingkungan eksekusi dan respons server. Pada pengujian ini, metode 1 (*Requests*) terbukti cukup untuk memenuhi seluruh kebutuhan data tanpa mengalami pemblokiran, sehingga peran metode 2 (*Selenium*) sebagai mekanisme cadangan tidak teruji secara nyata dalam skenario ini. Kondisi ini mengindikasikan bahwa desain redundansi dua metode tetap relevan dan diperlukan sebagai antisipasi risiko, meskipun pada praktiknya kebutuhan aktivasi metode cadangan bergantung pada seberapa ketat mekanisme anti-bot yang diterapkan oleh Google Scholar pada waktu tertentu, yang dapat berubah-ubah dan tidak selalu dapat diprediksi sebelumnya.

Secara keseluruhan, hasil penelitian ini menunjukkan bahwa pendekatan *scraping* berbasis dua metode yang saling melengkapi, disertai proses *parsing* berbasis *regex* dan evaluasi akurasi yang terukur, mampu menghasilkan dataset artikel ilmiah yang layak digunakan untuk kebutuhan analisis lebih lanjut, seperti pemetaan tren penelitian, analisis bibliometrik, maupun studi literatur sistematis pada topik pengembangan sistem informasi akademik.

4. PENUTUP

Meskipun eksekusi program berjalan dan mendapatkan hasil, tetapi pada sistem ini terdapat beberapa keterbatasan yang perlu diperhatikan. Pertama, Google Scholar menerapkan mekanisme CAPTCHA yang dapat aktif jika frekuensi *Requests* dinilai terlalu tinggi, sehingga sistem perlu dilengkapi dengan mekanisme penanganan CAPTCHA. Kedua, pada eksekusi yang dianalisis, Metode 2 (*Selenium*) tidak berkontribusi terhadap dataset akhir, sehingga efektivitas mekanisme redundansi dua metode terhadap skenario pemblokiran (rate-limit/CAPTCHA) belum sepenuhnya teruji dan perlu divalidasi kembali melalui eksekusi berulang pada kondisi jaringan atau waktu yang berbeda. Ketiga, struktur *HTML* Google Scholar dapat berubah sewaktu-waktu mengikuti pembaruan dari Google, sehingga kode *parser* perlu disesuaikan kembali secara berkala. Keempat, data abstrak yang diperoleh hanya merupakan cuplikan dan bukan teks penuh dari abstrak artikel. Kelima, belum adanya antarmuka pengguna grafis(GUI),sistem saat ini hanya dapat dioperasikan melalui antarmuka berbasis terminal, sehingga penggunaanya masih terbatas bagi pengguna yang memiliki pengetahuan teknis di bidang pemrograman.

DAFTAR PUSTAKA

Alfiansyah, T. R., Hidayat, A. A., Pratama, A. H., Aqshol, A., & Rafly, M. (2026). *Analisis Sentimen Kontaminasi Radioaktif di Kawasan Industri Cikande Menggunakan Algoritma Support Vector Machine*. *12*(1), 8–16.

Amanny Ulfah Nabiylah Ramadhanty, & Ina Najiyah. (2023). Implementasi Web Scraping Pada Situs Jurnal SintaMenggunakan Framework Selenium Webdriver Python. *JIKA (Jurnal Informatika)*, *7*(1), 29–36.

Arief, M. I., & Kurniawan, R. (2020). Pengembangan Sistem Aplikasi Web Scraper Harga Komoditas Menggunakan Metode Design Oriented Research. *Jambura Journal of Informatics*, *2*(1). https://doi.org/10.37905/jji.v2i1.4474

Brown, M. A., Gruen, A., Maldoff, G., Messing, S., Sanderson, Z., & Zimmer, M. (2025). Web scraping for research: Legal, ethical, institutional, and scientific considerations. *Big Data and Society*, *12*(4), 1–43. https://doi.org/10.1177/20539517251381686

Falah, Z. F., & Suryawan, S. T. F. (2021). *Sistem Rekomendasi Pemilihan Dosen Pembimbing Tugas Akhir Dengan Metrik Cosine Similarity*. http://eprints.ums.ac.id/id/eprint/95928%0Ahttp://eprints.ums.ac.id/95928/1/NASKAH PUBLIKASI - L200170149 %285%29.pdf

Glez-Peña, D., Lourenço, A., López-Fernández, H., Reboiro-Jato, M., & Fdez-Riverola, F. (2013). Web scraping technologies in an API world. *Briefings in Bioinformatics*, *15*(5), 788–797. https://doi.org/10.1093/bib/bbt026

Hafiz, Y. A., & Sudarmilah, E. (2023). Implementasi Web Scraping Pada Portal Berita Online. *Jurnal Ilmu Komputer*, *12*(1), 55–60.

Josi, A., Abdillah, L. A., Studi, P., Informatika, T., Komputer, F. I., Darma, U. B., Studi, P.,

Informasi, S., Komputer, F. I., Darma, U. B., Ahmad, J., & No, Y. (n.d.). *Penerapan Teknik Web Scraping pada Mesin Pencari Artikel Ilmiah*.

Khder, M. A. (2021). Web scraping or web crawling: State of art, techniques, approaches and application. *International Journal of Advances in Soft Computing and Its Applications*, *13*(3), 144–168. https://doi.org/10.15849/ijasca.211128.11

Kusumo, S. (2022). Penerapan Web Scraping Deskripsi Produk Menggunakan Selenium Python dan Framework Laravel. *JATISI (Jurnal Teknik Informatika Dan Sistem Informasi)*, *9*(4), 3426–3435. https://doi.org/10.35957/jatisi.v9i4.2727

Logos, K., Brewer, R., Langos, C., & Westlake, B. (2023). Establishing a framework for the ethical and legal use of web scrapers by cybercrime and cybersecurity researchers: learnings from a systematic review of Australian research. *International Journal of Law and Information Technology*, *31*(3), 186–212. https://doi.org/10.1093/ijlit/eaad023

Mitchell, R. (2018). *Ryan Mitchell Web Scraping with Python*. www.allitebooks.com

Mitra, V., Sujaini, H., Arif, B., & Negara, P. (2017). Rancang Bangun Aplikasi Web Scraping untuk Korpus Paralel Indonesia - Inggris dengan Metode HTML DOM. *Jurnal Sistem Dan Teknologi Informasi (JUSTIN)*, *5*(1), 1–6.

Muhammad Adhit Dwi Yuda. (2025). Implementasi Web Scraping Untuk Ekstraksi Data Penjual dan Produk Panel Surya Di E-Commerce. *KONSTELASI: Konvergensi Teknologi Dan Sistem Informasi*, *5*(1), 1–12. https://doi.org/10.24002/konstelasi.v5i1.11751

Ramadan, S. R., & Handaga, B. (2019). Forlap Scraper menggunakan Aplikasi Android. *Emitor: Jurnal Teknik Elektro*, *20*(1), 19–25. https://doi.org/10.23917/emitor.v20i1.8434

Richardson, L. (2016). Beautiful Soup Documentation. *Media.Readthedocs.Org*, 1–72. https://media.readthedocs.org/pdf/beautiful-soup-4/latest/beautiful-soup-4.pdf%0Ahttp://www.crummy.com/software/BeautifulSoup/bs4/doc/

Rifka Alkhilyatul Ma'rifat, I Made Suraharta, I. I. J. (2024). *Buku Ajar Imu Data* (Vol. 2).

Rizquina, A. Z., & Ratnasari, C. I. (2023). Implementasi Web Scraping untuk Pengambilan Data Pada Website E-Commerce. *Jurnal Teknologi Dan Sistem Informasi Bisnis*, *5*(4), 377–383. https://doi.org/10.47233/jteksis.v5i4.913

Satriajati, S., Bagus Panuntun, S., & Pramana, S. (2020). Implementasi Web Scraping Dalam Pengumpulan Berita Kriminal Pada Masa Pandemi COVID-19 Studi Kasus: Situs Berita detik.com. *Seminar Nasional Official Statistics 2020: Statistics in the New Normal: A Challenge of Big Data and Official Statistics*, 300–308.

Triandini, E., Jayanatha, S., Indrawan, A., Putra, G. W., & Iswara, B. (2019). *Metode Systematic Literature Review untuk Identifikasi Platform dan Metode Pengembangan Sistem Informasi di Indonesia*. *1*(2).

Wijaya, F. B., & Budiaji, W. (2025). Web Scraping Analysis of Job Platform Adoption in Banten's Manufacturing Sector. *RIGGS: Journal of Artificial Intelligence and Digital Business*, *4*(3), 2225–2231. https://doi.org/10.31004/riggs.v4i3.2282

Wijayanti, P. T., & Kharisma, D. B. (2024). Web Scraping dalam Aplikasi ChatGPT oleh Chatbot Berbasis Artificial Intelligence Berdasarkan Undang-Undang Nomor 28 Tahun 2014 Tentang Hak Cipta. *Souvereignty*, *3*(2), 114–121. https://botpress.com/id/blog/does-chatgpt-