PENGEMBANGAN SISTEM CERDAS UNTUK MANAJEMEN, ANALISIS, DAN OTOMATISASI DOKUMEN BERBASIS WEB

SUBJUDUL:

Analisis Teks Dokumen Administratif untuk Peringkasan Menggunakan NER-NLP

SKRIPSI

Dibuat untuk Melengkapi Syarat-Syarat yang Diperlukan untuk Memperoleh Diploma Empat Politeknik

Muhammad Nathan Sunarto
2207412024
TI-CCIT-8A

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER
POLITEKNIK NEGERI JAKARTA
2026

PENGEMBANGAN SISTEM CERDAS UNTUK MANAJEMEN, ANALISIS, DAN OTOMATISASI DOKUMEN BERBASIS WEB

SUBJUDUL:

Analisis Teks Dokumen Administratif untuk Peringkasan Menggunakan NER-NLP

SKRIPSI

Muhammad Nathan Sunarto
2207412024

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER
POLITEKNIK NEGERI JAKARTA
2026

SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertanda tangan di bawah ini:

Nama	:	Muhammad Nathan Sunarto
NIM	:	2207412024
Jurusan/Program Studi	:	Teknik Informatika dan Komputer / Teknik Informatika
Judul Skripsi	:	PENGEMBANGAN SISTEM CERDAS UNTUK MANAJEMEN, ANALISIS, DAN OTOMATISASI DOKUMEN BERBASIS WEB
Sub Judul	:	Analisis Teks Dokumen Administratif untuk Ringkasan Menggunakan NER-NLP

Menyatakan dengan sebenarnya bahwa skripsi ini benar-benar merupakan hasil karya saya sendiri, bebas dari peniruan terhadap karya dari orang lain. Kutipan pendapat dan tulisan orang lain ditunjuk sesuai dengan cara-cara penulisan karya ilmiah yang berlaku.

Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa dalam skripsi ini terkandung ciri-ciri plagiat dan bentuk-bentuk peniruan lain yang dianggap melanggar peraturan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Depok, 08 Juni 2026

Yang membuat pernyataan,

Muhammad Nathan Sunarto

NIM 2207412024

SURAT PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Politeknik Negeri Jakarta, saya bertanda tangan di bawah ini:

Nama : Muhammad Nathan Sunarto

NIM : 2207412024

Jurusan/Program Studi : T. Informatika dan Komputer/Teknik Informatika

Demi pengembangan ilmu pengetahuan, menyetuju untuk memberikan kepada Politeknik Negeri Jakarta Hak Bebas Royalti Non-Eksklusif atas karya ilmiah saya yang berjudul:

Analisis Teks Dokumen Administratif untuk Peringkasan Menggunakan NER-NLP.

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non Eksklusif ini, Politeknik Negeri Jakarta berhak menyimpan, mengalihmediakan/formatkan, mengelola dalam bentuk pangakalan data (database), merawat, dan mempublikasikan skripsi saya tanpa meminta izin dari saya selama tetap mencatumkan nama saya sebagai penulis/pencipta dan sebagai pemilih Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Depok, 30 Juni 2026

Yang membuat pernyataan

Muhammad Nathan Sunarto

NIM.2207412024

LEMBAR PENGESAHAN

Skripsi diajukan oleh:

Nama	: Muhammad Nathan Sunarto
NIM	: 2207412024
Program Studi	: Teknik Informatika
Judul Skripsi	: Pengembangan Sistem Cerdas Untuk Manajemen, Analisis, Dan Otomatisasi Dokumen Berbasis Web
Judul Skripsi	: Analisis Dokumen Administratif Untuk Peringkasan Menggunakan NER-NLP

Pada hari ini Selasa Tanggal 23, bulan Juni tahun 2026, telah diadakan Sidang Skripsi untuk saudara Muhammad Nathan Sunarto dan dinyatakan **LULUS**

Disahkan oleh

Pembimbing I	: Risna Sari, S.Kom., M.T.I..	( )
Penguji I	: Dr. Dewi Yanti Liliana, S.Kom., M.Kom.	( )
Penguji II	: Rizki Elisa Nalawati, S.T., M.T.	( )
Penguji III	: Ariawan Andi Suhandana, S.Kom., M.T.I.	( )

Mengetahui:
Jurusan Teknik Informatika dan Komputer
Ketua

Dr. Anita Hidayati, S.Kom., M.Kom.
NIP. 19790832003122003

Analisis Teks Dokumen Administratif untuk Identifikasi Bagian Teks Variabel Dinamis

ABSTRAK

Dokumen administratif merupakan salah satu sumber informasi penting yang digunakan dalam berbagai aktivitas organisasi dan institusi. Namun, proses identifikasi informasi penting pada dokumen administratif masih banyak dilakukan secara manual sehingga membutuhkan waktu yang cukup lama dan berpotensi menimbulkan kesalahan. Penelitian ini bertujuan mengembangkan sistem analisis dokumen administratif yang mampu melakukan ekstraksi informasi penting dan menghasilkan ringkasan dokumen secara otomatis. Sistem yang dikembangkan memanfaatkan PaddleOCR untuk mengekstraksi teks dari dokumen PDF digital, PDF hasil pemindaian, dan DOCX, serta model IndoBERT yang telah melalui proses fine-tuning untuk melakukan Named Entity Recognition (NER) terhadap entitas administratif seperti nomor surat, jenis dokumen, tanggal, pengirim, penerima, perihal, lokasi, waktu, dan isi dokumen. Selain itu, sistem mengimplementasikan LayoutLMv3 untuk mendukung ekstraksi informasi pada dokumen yang mengandung tabel dan menghasilkan ringkasan dokumen berdasarkan hasil identifikasi entitas. Model NER dilatih melalui 13 skenario eksperimen dengan berbagai strategi optimasi, dan konfigurasi terbaik diperoleh pada Run ID #13 dengan nilai F1-score sebesar 88,56%. Sistem kemudian diimplementasikan dalam bentuk REST API menggunakan FastAPI dan di-deploy pada AWS EC2. Hasil pengujian menunjukkan bahwa sistem mampu mengidentifikasi informasi penting secara otomatis dan menghasilkan ringkasan dokumen yang relevan, sehingga dapat membantu meningkatkan efisiensi pengelolaan dan analisis dokumen administratif.

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Allah SWT atas segala rahmat, hidayah, dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Analisis Dokumen Administratif untuk Identifikasi Teks Variabel Dinamis Menggunakan OCR dan Named Entity Recognition Berbasis IndoBERT" dengan baik dan tepat waktu.

Skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Terapan (S.Tr.Kom.) pada Program Studi Teknik Informatika, Jurusan Teknik Informatika dan Komputer, Politeknik Negeri Jakarta. Penelitian ini berfokus pada pengembangan sistem yang mampu melakukan ekstraksi teks dari dokumen administratif menggunakan PaddleOCR, mengidentifikasi entitas penting seperti nomor surat, jenis dokumen, tanggal, pengirim, penerima, perihal, dan lokasi menggunakan model IndoBERT yang telah melalui proses fine-tuning, serta menghasilkan ringkasan dokumen secara otomatis melalui layanan REST API yang diimplementasikan pada lingkungan AWS EC2. Sistem ini diharapkan dapat membantu meningkatkan efisiensi proses analisis dokumen administratif serta mengurangi waktu yang dibutuhkan pengguna dalam memahami informasi penting yang terkandung dalam dokumen.

Dalam proses penyusunan skripsi ini, penulis menyadari bahwa keberhasilannya tidak terlepas dari dukungan, bimbingan, dan bantuan berbagai pihak. Oleh karena itu, pada kesempatan ini penulis menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Ibu Dr. Anita Hidayati, S.Kom., M.Kom., Ketua Jurusan Teknik Informatika dan Komputer Politeknik Negeri Jakarta, atas dukungan dan arahan yang diberikan kepada seluruh mahasiswa.
2. Ibu Euis Oktavianti, S.Si., M.T.I., Kepala Program Studi Teknik Informatika Politeknik Negeri Jakarta, atas bimbingan dan fasilitas akademik yang menunjang proses penelitian.

3. Ibu Risna Sari sebagai Dosen Pembimbing, yang telah meluangkan waktu, tenaga, dan pikiran untuk memberikan bimbingan, arahan, serta masukan yang sangat berharga selama proses penelitian dan penulisan skripsi ini.

4. Seluruh Dosen dan Staf Akademik Jurusan Teknik Informatika dan Komputer Politeknik Negeri Jakarta, atas ilmu dan pengalaman yang telah diberikan kepada penulis selama masa perkuliahan.

5. Kedua orang tua dan seluruh keluarga penulis, yang senantiasa memberikan doa, semangat, dukungan moral, dan materi yang tidak ternilai selama penulis menempuh pendidikan.

6. Seluruh rekan mahasiswa seperjuangan angkatan 2022, atas kebersamaan, diskusi, dan semangat yang selalu menginspirasi penulis selama masa studi.

7. Seluruh jajaran team FC Barcelona yang telah menghibur, memberikan semangat serta dukungan dengan cara memenangkan liga.

8. Semua pihak yang telah membantu dan mendukung yang tidak dapat penulis sebutkan satu per satu.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna. Oleh karena itu, penulis mengharapkan kritik dan saran yang bersifat membangun demi penyempurnaan skripsi ini di masa mendatang. Besar harapan penulis agar skripsi ini dapat memberikan manfaat bagi pembaca, khususnya bagi pengembangan ilmu pengetahuan di bidang Teknik Informatika dan Rekayasa Perangkat Lunak.

Jakarta, 9 Juni 2026

Muhammad Nathan Sunarto

NIM 2207412024

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1. Latar Belakang Masalah

Dokumen administratif merupakan salah satu media utama yang digunakan oleh institusi pendidikan, pemerintahan, maupun organisasi dalam menjalankan proses administrasi dan komunikasi formal. Berbagai jenis dokumen seperti surat undangan, surat permohonan, surat keterangan, nota dinas, pengumuman, serta dokumen akademik lainnya mengandung informasi penting yang digunakan sebagai dasar pengambilan keputusan dan pelaksanaan kegiatan organisasi. Informasi tersebut umumnya berupa nomor surat, tanggal dokumen, pengirim, penerima, perihal, lokasi kegiatan, dan jenis dokumen. Keakuratan dalam mengidentifikasi informasi tersebut sangat penting karena berkaitan langsung dengan validitas dokumen dan efektivitas proses administrasi.

Seiring dengan meningkatnya transformasi digital di berbagai institusi, jumlah dokumen administratif yang dihasilkan dan diproses setiap hari terus bertambah. Pemerintah Indonesia melalui kebijakan Sistem Informasi Kearsipan Nasional (SKN) 2025-2029 juga mendorong pengintegrasian sistem informasi kearsipan dengan Sistem Pemerintahan Berbasis Elektronik (SPBE), yang semakin mempercepat volume digitalisasi dokumen administrasi. Meskipun sebagian besar dokumen telah tersedia dalam format digital, proses identifikasi informasi penting masih banyak dilakukan secara manual. Staf administrasi sering kali harus membaca keseluruhan isi dokumen untuk menemukan informasi yang diperlukan sebelum melakukan pengarsipan, pelaporan, atau penyusunan tindak lanjut. Proses tersebut membutuhkan waktu yang relatif lama, terutama ketika jumlah dokumen yang harus diproses sangat banyak. Selain itu, proses manual juga berpotensi menimbulkan kesalahan identifikasi informasi akibat kelelahan maupun perbedaan interpretasi pengguna terhadap isi dokumen. Penelitian oleh (A. Yani 2024) menunjukkan bahwa sistem otomatisasi berbasis web mampu meningkatkan efisiensi pekerjaan administrasi secara signifikan dibandingkan dengan proses manual.

Permasalahan lain yang sering ditemukan adalah keberagaman format dokumen administratif. Meskipun memiliki struktur yang relatif serupa, setiap jenis dokumen dapat memiliki tata letak, gaya penulisan, serta susunan informasi yang berbeda. Variasi tersebut menyebabkan proses ekstraksi informasi secara otomatis menjadi lebih kompleks karena sistem harus mampu mengenali pola informasi yang muncul pada berbagai bentuk dokumen. Pada dokumen hasil pemindaian (scanned document), tantangan menjadi lebih besar karena informasi yang tersedia masih berbentuk citra dan perlu dikonversi terlebih dahulu menjadi teks yang dapat diproses oleh komputer.

Perkembangan teknologi Artificial Intelligence (AI), khususnya pada bidang Natural Language Processing (NLP) dan Computer Vision, membuka peluang untuk mengatasi permasalahan tersebut. sistem manajemen dokumen adaptif yang dapat belajar dari struktur dokumen baru secara terus-menerus untuk meningkatkan efisiensi pengelolaan dokumen dalam jangka panjang. Salah satu teknologi yang banyak digunakan adalah Optical Character Recognition (OCR), yaitu teknologi yang mampu mengubah teks pada gambar atau dokumen hasil pemindaian menjadi teks digital yang dapat diproses lebih lanjut. Setelah teks berhasil diekstraksi, proses identifikasi informasi penting dapat dilakukan menggunakan teknik Named Entity Recognition (NER), yaitu salah satu tugas dalam NLP yang bertujuan untuk mengenali dan mengklasifikasikan entitas tertentu ke dalam kategori yang telah ditentukan, seperti nama orang, organisasi, lokasi, tanggal, maupun informasi administratif lainnya. Selain itu, teknologi text summarization dapat dimanfaatkan untuk menghasilkan ringkasan dokumen secara otomatis sehingga pengguna dapat memahami isi dokumen dengan lebih cepat dan efisien.

Penelitian mengenai ekstraksi informasi dokumen menggunakan pendekatan OCR dan NER telah menunjukkan hasil yang menjanjikan dalam membantu proses otomatisasi dokumen. Sebuah penelitian pada tahun 2025 berhasil mengimplementasikan sistem OCR dan NLP menggunakan PyTesseract dan model fine-tuned IndoBERT untuk pengelolaan dokumen administratif, yang mencapai akurasi klasifikasi 93,6% dan F1-Score NER sebesar 0,989. Namun, sebagian besar penelitian masih berfokus pada identifikasi entitas umum dan

belum secara khusus menangani kebutuhan identifikasi teks variabel dinamis pada dokumen administratif berbahasa Indonesia. Padahal, informasi seperti nomor surat, perihal, pengirim, penerima, dan jenis dokumen merupakan komponen penting yang sering digunakan dalam proses administrasi dan pengarsipan dokumen. Penelitian oleh Ramdhani (2025) menunjukkan bahwa model deep learning untuk NER berbahasa Indonesia belum sepenuhnya efektif dalam menangani kekhususan struktur dokumen administratif yang semi-terstruktur, sehingga diperlukan pendekatan hybrid yang menggabungkan deep learning dengan rule-based untuk mencapai akurasi yang lebih tinggi. Pendekatan hybrid yang dikembangkan oleh Ramdhani (2025) berhasil mencapai skor F1 rata-rata 98% pada sepuluh jenis dokumen kepegawaian.

Berdasarkan permasalahan tersebut, penelitian ini mengusulkan pengembangan sistem analisis dokumen administratif berbasis OCR dan Named Entity Recognition menggunakan model transformer IndoBERT. Sistem yang dikembangkan mampu menerima dokumen administratif dalam format PDF maupun dokumen hasil pemindaian, melakukan ekstraksi teks menggunakan OCR, mengidentifikasi teks variabel dinamis melalui model NER, serta menghasilkan ringkasan dokumen secara otomatis. Dengan adanya sistem ini, diharapkan proses identifikasi informasi penting pada dokumen administratif dapat dilakukan secara lebih cepat, akurat, dan efisien sehingga mampu membantu pekerjaan administrasi serta mendukung pengelolaan dokumen digital di lingkungan institusi.

1.2. Perumusan Masalah

1. Bagaimana merancang pipeline yang mampu mengekstrak teks dan informasi tata letak dari dokumen PDF administratif menggunakan PyMuPDF, serta mengidentifikasi bagian-bagian struktural dokumen (header, body, tabel, tanda tangan) berdasarkan fitur spasial dan tekstual?

2. Bagaimana membangun model Named Entity Recognition (NER) yang dapat mengenali entitas-entitas kunci dalam dokumen administratif, yaitu NOMOR_SURAT, JENIS_DOKUMEN, TANGGAL, PENGIRIM, PENERIMA, PERIHAL, dan LOKASI?

(Penelitian terkini menunjukkan bahwa model transformer seperti IndoBERT

dapat di-fine-tuning untuk mengenali entitas pada teks berbahasa Indonesia dengan performa yang baik [Muchtar dkk., 2025; Ramdhani, 2025].)

3. Bagaimana menghasilkan ringkasan otomatis dari isi dokumen (bagian body) yang informatif dan akurat?

4. Bagaimana mengevaluasi kinerja sistem dalam mengekstraksi entitas dan menghasilkan ringkasan dibandingkan dengan anotasi manual?

1.3. Batasan Masalah

- Jenis dan format dokumen: Dokumen administratif dalam format PDF yang berasal dari lingkungan JTIK, meliputi Surat Tugas, Nota Dinas, Surat Undangan, dan Surat Keputusan. Diasumsikan dokumen lahir digital (bukan hasil pindaian) sehingga teks dapat diekstrak langsung menggunakan PyMuPDF.
- Bahasa dokumen: Bahasa Indonesia.
- Entitas yang diekstrak: Terbatas pada header Header(teks kop instansi), Body(teks isi utama), Table (area tabel), Signature (area tanda tangan),
- Pendekatan NER: Menggunakan pustaka IndoBERT dengan model bahasa Indonesia yang telah dilatih ulang (fine-tuning) atau menggunakan pendekatan berbasis aturan untuk entitas tertentu. Penelitian oleh Muchtar dkk. (2025) menunjukkan bahwa pendekatan fine-tuning IndoBERT dapat menghasilkan F1-score yang stabil untuk pengenalan entitas seperti PER, ORG, LOC, DATE, dan CRD pada teks berbahasa Indonesia.
- Metode peringkasan: Menggunakan metode ekstraktif sederhana (Template-based/Rulebased) karena keterbatasan data latih untuk model abstraktif.
- Output sistem: Berupa file JSON yang berisi entitas yang ditemukan beserta posisinya (bounding box) dan ringkasan teks.

.

1.4. Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah:

1. Membangun pipeline otomatis untuk ekstraksi teks dan tata letak dari dokumen PDF administratif menggunakan PyMuPDF.
2. Mengembangkan modul identifikasi bagian struktural dokumen (header, body, tabel, tanda tangan) berdasarkan informasi spasial.
3. Melatih dan mengevaluasi model NER untuk mengenali entitas-entitas administratif pada teks dokumen.
4. Mengimplementasikan modul peringkasan teks untuk menghasilkan ringkasan isi dokumen.
5. Mengukur akurasi ekstraksi entitas dan kualitas ringkasan yang dihasilkan.

1.5. Manfaat Penelitian

Manfaat penelitian ini dibedakan menjadi dua aspek:

1.5.1 Manfaat Teoritis

1. Memberikan kontribusi pada pengembangan sistem pemahaman dokumen berbahasa Indonesia, khususnya untuk domain administratif.
2. Menjadi referensi bagi peneliti lain dalam mengintegrasikan OCR, NER, dan peringkasan teks pada dokumen semi-terstruktur.

1.5.2 Manfaat Praktis

1. Memudahkan staf administrasi JTIK dalam mencari dan memanfaatkan informasi dari arsip dokumen.
2. Mengurangi waktu dan potensi kesalahan dalam identifikasi informasi penting pada dokumen administratif.
3. Menjadi dasar pengembangan sistem manajemen dokumen yang lebih cerdas di lingkungan institusi.

1.6. Sistematika Penulisan

Laporan penelitian ini disusun dalam lima bab yang saling berkaitan untuk memberikan gambaran yang utuh dan runtut mengenai seluruh proses pengerjaan. Bagian awal adalah Bab I Pendahuluan, yang berfungsi sebagai pengantar untuk menjelaskan alasan mengapa penelitian ini dilakukan, masalah apa yang ingin diselesaikan, serta tujuan dan batasan yang ditetapkan.

Pembahasan kemudian berlanjut ke Bab II Tinjauan Pustaka, yang akan mengupas teori-teori pendukung dan menelaah penelitian terdahulu yang relevan sebagai landasan berpikir. Setelah bekal teori cukup, Bab III Metode Penelitian akan menjelaskan langkah-langkah teknis yang diambil, mulai dari perancangan sistem hingga strategi pengujian yang akan dilakukan.

Hasil nyata dari perancangan tersebut akan disajikan dalam Bab IV Hasil dan Pembahasan, yang menampilkan prototipe aplikasi yang telah jadi serta analisis kinerjanya. Akhirnya, laporan ditutup dengan Bab V Penutup, yang berisi kesimpulan menyeluruh dari penelitian serta saran-saran untuk pengembangan lebih lanjut di masa depan.

BAB II
TINJAUAN PUSTAKA

Bab ini mengkaji konsep-konsep teoritis dan hasil penelitian terdahulu yang relevan sebagai dasar dalam pengembangan sistem analisis teks dokumen administratif untuk identifikasi bagian teks variabel dinamis menggunakan pendekatan NER-NLP. Pembahasan difokuskan pada karakteristik dokumen administratif terstruktur, teknologi ekstraksi dan pemrosesan teks, metodologi identifikasi pola, serta kerangka kerja sistem pendukung otomatisasi.

2.1. Karakteristik Dokumen Administratif JTIK sebagai Sumber Data

Dokumen administratif di lingkungan JTIK memiliki keragaman jenis dan struktur. Beberapa jenis dokumen yang umum ditemui antara lain Surat Tugas yang berisi penugasan kepada dosen atau tenaga kependidikan, mencakup nomor surat, dasar penugasan, nama yang ditugaskan, tujuan, waktu, dan tempat; Nota Dinas yang digunakan untuk komunikasi internal, biasanya berisi permohonan, pemberitahuan, atau instruksi; Kwitansi sebagai bukti pembayaran, berisi nomor kwitansi, jumlah uang, keperluan, dan tanda tangan penerima; Surat Undangan yang mengundang pihak tertentu untuk menghadiri acara, mencakup hari atau tanggal, waktu, tempat, dan acara; serta Surat Keputusan yang berisi penetapan resmi, misalnya pengangkatan panitia. Struktur dokumen umumnya terdiri dari Header berupa kop surat yang memuat logo, nama instansi atau jurusan, alamat, dan lain-lain; Body yang merupakan isi utama, bisa berupa paragraf atau poin-poin; Table yang kadang-kadang terdapat untuk data terstruktur seperti daftar peserta atau rincian biaya; serta Signature area penandatanganan yang biasanya berisi jabatan, nama, dan NIP. Penelitian oleh Ramdhani (2025) menyoroti bahwa dokumen administratif memiliki karakteristik semi-terstruktur dengan entitas-entitas yang mengikuti pola tetap, sehingga membutuhkan pendekatan NER yang dapat menggabungkan kemampuan generalisasi dari deep learning dengan ketelitian dari metode rule-based. Keberagaman format dan tata letak menjadi tantangan dalam ekstraksi informasi otomatis, sehingga diperlukan pendekatan yang memanfaatkan informasi posisi teks.

2.2. Natural Language Processing (NLP)

Natural Language Processing (NLP) merupakan salah satu cabang dari Artificial Intelligence (AI) yang menggabungkan ilmu komputer dan linguistik untuk memungkinkan komputer memahami, menganalisis, serta menghasilkan bahasa alami yang digunakan manusia. Tujuan utama NLP adalah menjembatani komunikasi antara manusia dan komputer sehingga komputer dapat mengolah informasi yang disampaikan dalam bentuk bahasa sehari-hari. NLP terdiri atas dua komponen utama, yaitu Natural Language Understanding (NLU) yang berfungsi memahami makna dari bahasa manusia dan Natural Language Generation (NLG) yang bertugas menghasilkan teks atau kalimat yang dapat dipahami oleh manusia (Khurana et al., 2023).

Teknologi NLP telah diterapkan pada berbagai bidang, seperti penerjemahan bahasa (machine translation), deteksi spam email, ekstraksi informasi, peringkasan dokumen, sistem tanya jawab (question answering), chatbot, serta analisis sentimen. Penerapan tersebut menunjukkan bahwa NLP mampu membantu komputer dalam memahami isi dokumen maupun menghasilkan informasi yang relevan berdasarkan teks yang diproses (Khurana et al., 2023).

2.3. Named Entity Recognition (NER)

Named Entity Recognition (NER) merupakan salah satu tugas utama dalam Natural Language Processing (NLP) yang berfokus pada proses mengidentifikasi serta mengklasifikasikan entitas bernama (named entities) yang terdapat pada teks tidak terstruktur ke dalam kategori yang telah ditentukan. Entitas tersebut dapat berupa nama orang (person), organisasi (organization), lokasi (location), ekspresi waktu (temporal expression), nilai numerik (numerical values), maupun entitas khusus pada domain tertentu seperti gen, protein, dan penyakit dalam bidang biomedis. Tujuan utama NER adalah mendeteksi batas suatu entitas dalam teks sekaligus menentukan kategori semantiknya sehingga informasi penting dapat diekstraksi secara otomatis (Keraghel et al., 2024).

Perkembangan NER telah mengalami perubahan yang signifikan, dimulai dari pendekatan berbasis aturan (rule-based), kemudian berkembang ke metode machine learning, deep learning, hingga model berbasis Transformer dan Large Language Models (LLMs). Penggunaan model-model modern tersebut mampu meningkatkan kemampuan sistem dalam memahami konteks kalimat sehingga proses identifikasi entitas menjadi lebih akurat pada berbagai domain, seperti kesehatan, pencarian informasi, sistem tanya jawab, dan penerjemahan bahasa (Keraghel et al., 2024). Contoh dari penggunaan NER sebagai berikut:

```
"Presiden Joko Widodo menghadiri peresmian rumah sakit baru di Bandung pada 10 Januari 2025."
```

Pada kalimat tersebut, sistem NER akan mengenali beberapa entitas penting sebagai berikut.

Teks	Kategori NER
Joko Widodo	Person
Bandung	Location
10 Januari 2025	Date

Dalam contoh di atas, sistem NER akan membaca teks dan mengenali bahwa "Joko Widodo" merupakan nama seseorang sehingga diklasifikasikan sebagai Person. Selanjutnya, kata "Bandung" dikenali sebagai nama tempat sehingga dikategorikan sebagai Location. Adapun kata "10 Januari 2025" diidentifikasi sebagai informasi waktu sehingga dimasukkan ke kategori Date .

2.4. Ekstraksi Teks dan Tata Letak dari PDF dengan PyMuPDF dan PaddleOCR

Named Entity Recognition (NER) adalah subtugas NLP yang bertujuan untuk mengidentifikasi dan mengklasifikasikan entitas bernama dalam teks ke dalam kategori yang telah ditentukan, seperti orang, organisasi, lokasi, tanggal, waktu,

dan nominal. Dalam konteks dokumen formal, entitas yang penting meliputi nomor surat, tanggal penandatanganan, nama instansi, jabatan, dan nilai kontrak.

Pendekatan modern untuk NER umumnya menggunakan arsitektur berbasis transformer, seperti BERT dan variannya. Untuk teks berbahasa Indonesia, dapat digunakan model IndoBERT yang telah dilatih pada korpus bahasa Indonesia. Model ini mampu menangkap konteks kalimat dengan baik dan memberikan representasi token yang kaya, sehingga meningkatkan akurasi NER. Penelitian oleh Muchtar dkk. (2025) menunjukkan bahwa IndoBERT mampu memberikan performa yang konsisten dengan nilai F1-score yang stabil untuk identifikasi entitas pada teks berbahasa Indonesia. Model IndoBERT yang telah fine-tuned untuk tugas NER juga telah tersedia dalam bentuk pipeline yang kompatibel dengan HuggingFace, memungkinkan penggunaan yang lebih mudah untuk berbagai aplikasi NLP bahasa Indonesia.

Sebelum proses ekstraksi entitas dilakukan, dokumen berbentuk gambar atau hasil pindai perlu dikonversi terlebih dahulu menjadi teks menggunakan Optical Character Recognition (OCR). Salah satu framework OCR yang banyak digunakan adalah PaddleOCR. PaddleOCR merupakan pustaka berbasis deep learning yang mendukung deteksi area teks, pengenalan karakter, dan klasifikasi orientasi dokumen. Keunggulan PaddleOCR terletak pada kemampuannya mengenali teks multibahasa termasuk bahasa Indonesia, serta akurasi yang tinggi pada dokumen formal seperti surat, formulir, dan arsip administrasi. Versi terbaru, PaddleOCR-VL, yang diluncurkan pada tahun 2025, mendukung 109 bahasa dengan arsitektur vision-language model (VLM) berukuran 0,9B parameter yang mampu mengenali elemen kompleks seperti teks, tabel, formula, dan bagan secara akurat (Cui dkk., 2025). Dalam penelitian ini, PaddleOCR berperan sebagai tahap awal untuk mengekstraksi teks dari dokumen sebelum teks tersebut diproses lebih lanjut oleh model NER.

Penelitian sebelumnya menunjukkan bahwa kualitas ekstraksi entitas sangat dipengaruhi oleh representasi teks dan metode pelabelan. Dengan menggunakan model transformer yang di-fine-tuning pada dataset domain spesifik, performa NER dapat mencapai F1-score di atas 90% untuk entitas umum. Penelitian oleh

Ramdhani (2025) mengembangkan pendekatan hybrid yang menggabungkan beberapa model deep learning (IndoBERT, T5, Qwen, dan SahabatAI) dengan metode rule-based linguistik sebagai mekanisme label refinement, dan berhasil mencapai skor F1 rata-rata 98% pada sepuluh jenis dokumen kepegawaian. Kombinasi antara PaddleOCR untuk ekstraksi teks dan model transformer untuk NER memungkinkan sistem mampu melakukan identifikasi informasi penting secara otomatis dari dokumen administratif dengan tingkat akurasi yang lebih tinggi.

2.5. Analisis Tata Letak Dokumen

Analisis tata letak dokumen (Document Layout Analysis/DLA) bertujuan untuk mengidentifikasi dan mengklasifikasikan region-region dalam halaman dokumen, seperti blok teks, tabel, gambar, dan garis. Pendekatan tradisional menggunakan aturan heuristik berdasarkan koordinat dan ukuran font, sementara pendekatan modern memanfaatkan deep learning (misal: LayoutLM, DiT). Penelitian oleh BinMakhashen dan Mahmoud (2020) yang dikutip kembali dalam survei terbaru menunjukkan bahwa informasi urutan baca (reading order) sangat penting untuk merekonstruksi struktur dokumen. Karena keterbatasan data latih, penelitian ini akan menggunakan pendekatan heuristik sederhana. Header dideteksi sebagai blok teks di bagian atas halaman dengan font besar atau tebal, biasanya berisi nama instansi. Body adalah blok teks yang membentuk paragraf, terletak di tengah halaman. Table adalah region yang mengandung teks tersusun dalam baris dan kolom, dapat dideteksi dengan kerapatan spasial. Signature adalah region di bagian bawah yang mengandung teks seperti "Ketua Jurusan" atau "a.n." dan biasanya dikelilingi ruang kosong.

Model modern seperti LayoutLMv3, yang dikembangkan oleh Microsoft Research, menyatukan teks, tata letak, dan gambar dalam satu arsitektur transformer terpadu dengan menghilangkan hambatan CNN, sehingga menjadikannya salah satu model transformer dokumen paling efisien dan kuat yang tersedia. Penelitian oleh Liu, Shi dan Chen (2025) menunjukkan bahwa pendekatan berbasis transformer untuk pemrosesan dokumen tingkat wacana dapat memanfaatkan pengetahuan bahasa sebelumnya dan penguraian hierarkis. Dalam pipeline ini, urutan baca akan ditentukan berdasarkan koordinat vertikal dan horizontal.

2.6. Named Entity Recognition (NER) untuk Entitas Administratif

NER adalah subtugas NLP yang bertujuan mengidentifikasi dan mengklasifikasikan entitas bernama dalam teks ke dalam kategori yang telah ditentukan. Untuk bahasa Indonesia, telah tersedia model terlatih seperti id_core_news_sm dari spaCy atau model berbasis Transformer seperti

IndoBERT. Namun, model umum tersebut tidak mencakup entitas khusus administratif, misalnya NOMOR_SURAT, JENIS_DOKUMEN, sehingga diperlukan pelatihan ulang (fine-tuning) atau pembuatan pattern matching berbasis aturan. Penelitian oleh Al Faraby dkk. (2020) yang dikutip kembali dalam studi tahun 2025 tentang pengenalan entitas peristiwa pada berita online menunjukkan bahwa arsitektur BiLSTM-CNN dapat mencapai rata-rata F1-score sebesar 81%. Namun, untuk dokumen administratif, pendekatan yang digunakan dalam penelitian ini adalah kombinasi aturan berbasis pola regex untuk entitas dengan format tetap (nomor surat, tanggal) dan model machine learning (CRF atau spaCy) untuk entitas yang lebih kontekstual (pengirim, penerima, perihal, lokasi), dilatih menggunakan data anotasi manual dari dokumen JTIK. Penelitian yang mengimplementasikan Stanford NER untuk pemberian entitas pada dokumen Bahasa Indonesia menunjukkan bahwa pendekatan berbasis CRF yang sudah dikembangkan untuk bahasa Inggris dapat diadaptasi untuk dokumen berbahasa Indonesia dengan hasil yang cukup baik.

Studi oleh Muchtar dkk. (2025) mengonfirmasi bahwa pendekatan fine-tuning IndoBERT untuk tugas NER pada teks berbahasa Indonesia efektif dan menghasilkan performa yang stabil, dengan F1-score yang konsisten di seluruh skema validasi. Penelitian oleh Ramdhani (2025) juga menunjukkan bahwa integrasi metode rule-based ke dalam sistem NER berbasis deep learning dapat secara signifikan meningkatkan akurasi pengelolaan dokumen kepegawaian di lingkungan pemerintahan. Karena teks dari PyMuPDF bebas dari kesalahan OCR, akurasi NER diharapkan lebih tinggi.

2.7. Penggunaan Library Pada Project

2.7.1 Web Framework

- **FastAPI**

  Framework Python untuk membangun REST API yang cepat, mendukung asynchronous programming, validasi data otomatis menggunakan Pydantic, serta dokumentasi API interaktif.

- **Uvicorn**

ASGI (Asynchronous Server Gateway Interface) server yang digunakan untuk menjalankan aplikasi FastAPI dengan dukungan asynchronous sehingga mampu menangani banyak permintaan secara efisien.

- **Python-Multipart**

Library untuk memproses data bertipe multipart/form-data, seperti upload file atau pengiriman form melalui HTTP.

2.7.2 Natural Language Processing

- **Transformers**

Transformers merupakan library yang dikembangkan oleh Hugging Face untuk menyediakan implementasi berbagai model berbasis arsitektur Transformer, seperti BERT, RoBERTa, GPT, T5, DistilBERT, dan IndoBERT. Library ini menyediakan model yang telah melalui proses pre-training sehingga dapat langsung digunakan untuk berbagai tugas NLP atau dilakukan fine-tuning pada dataset tertentu.

Pada penelitian ini, library Transformers digunakan untuk memuat model IndoBERT Base P1 yang kemudian di-fine-tuning pada tugas Named Entity Recognition (NER) untuk mengidentifikasi bagian teks variabel dinamis pada dokumen administratif. Library ini juga menyediakan tokenizer yang bertugas mengubah kalimat menjadi token beserta indeks numeriknya sebelum diproses oleh model. Selain itu, Transformers menyediakan API yang terintegrasi dengan PyTorch sehingga proses pelatihan (training), validasi, maupun inferensi dapat dilakukan secara efisien.

Penggunaan transformers meliputi beberapa proses yaitu:

A. Self attention

Self-Attention merupakan mekanisme utama pada arsitektur Transformer yang memungkinkan setiap token memperhatikan hubungan dengan seluruh token lainnya dalam satu kalimat. Dengan mekanisme ini, model dapat memahami konteks secara menyeluruh tanpa harus memproses kata secara berurutan seperti pada RNN. Sebagai Contoh kalimat yaitu:

```
Surat tugas diberikan kepada mahasiswa.
```

ketika model memproses kata "Surat", model juga memperhatikan hubungan kata tersebut dengan "tugas", "diberikan", dan "mahasiswa", sehingga representasi kata yang dihasilkan menjadi lebih kontekstual.

Bentuk persamaan Self attention sebagai berikut:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Dimana penjelasannya adalah:

Q (Query)	representasi token yang sedang diproses.
K (Key)	representasi seluruh token.
V (Value)	informasi yang akan dipelajari model.
dk	dimensi vektor Key.

B. Softmax

Nilai hasil perkalian Query dan Key belum berupa probabilitas. Oleh karena itu, digunakan fungsi Softmax untuk mengubah skor perhatian (attention score) menjadi distribusi probabilitas. Bentuk persamaan softmax sebagai berikut:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

Yang dari persamaan itu menghasilkan Contoh:

```
[2.5, 1.2, 0.3]
```

Setelah melalui Softmax menjadi

```
[0.70, 0.22, 0.08]
```

Yang artinya hasil dari softmax menunjukan token pertama memperoleh perhatian terbesar dibanding token lainnya.

C. Positional Encoding

Transformer memproses seluruh token secara paralel, model tidak mengetahui urutan kata secara alami. Oleh karena itu ditambahkan Positional Encoding agar model dapat mengenali posisi setiap token di dalam kalimat.

Persamaan Positional Encoding yaitu:

$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d}}\right)$ dan $PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d}}\right)$

pos	Posisi Token
i	Indeks Dimensi Embedding
D	Ukuran Dimensi Embedding

Embedding suatu token tidak hanya terdiri dari satu angka, tetapi berupa vektor. Misalnya pada BERT Base (termasuk IndoBERT Base), satu token direpresentasikan oleh 768 dimensi. Maka penggunaan 2 persamaan itu adalah penghitungan setiap dimensi secara bergantian dengan fungsi SIN dan COS.

- **Pytorch**

PyTorch merupakan framework deep learning yang dikembangkan oleh Meta AI untuk membangun, melatih, dan melakukan inferensi pada model jaringan saraf (neural network). Framework ini mendukung automatic differentiation (autograd) sehingga proses perhitungan gradien dapat dilakukan secara otomatis selama proses pelatihan.

PyTorch digunakan sebagai framework utama dalam proses fine-tuning model IndoBERT. Seluruh proses forward propagation, perhitungan loss, backpropagation, hingga pembaruan parameter model dilakukan menggunakan

PyTorch. Selain itu, PyTorch mendukung pemanfaatan GPU sehingga waktu pelatihan model menjadi lebih efisien.

Proses Penghitungan pada Pytorch Sebagai berikut:

A. Forward Propagation

proses ketika data masukan (input) diteruskan melalui seluruh lapisan (layer) pada model neural network untuk menghasilkan prediksi. Persamaan yang di gunakan forward propagation sebagai berikut

$$\hat{y} = f(x)$$

B. Cross-Entropy Loss

Hasil prediksi dibandingkan dengan label sebenarnya menggunakan fungsi Cross Entropy Loss yang umum digunakan pada tugas klasifikasi token. Persamaan yang di gunakan pada proses ini adalah:

$$L = -\sum_{i=1}^{C} y_i \log(p_i)$$

C	Jumlah kelas
yi	Label real
pi	Probabilitas prediksi model

Dimana dari hasil penghitungan dinyatakan bahwa semakin kecil nilai loss yang terjadi maka semakin baik performa model.

C. Backpropagation

Setelah nilai loss diperoleh, PyTorch menghitung gradien setiap parameter menggunakan algoritma backpropagation. Parameter model diperbarui dengan persamaan:

$$\theta = \theta - \eta \frac{\partial L}{\partial \theta}$$

θ	Parameter Model
η	Learning Rate
$\frac{\partial L}{\partial \theta}$	Gradien Terhadap Parameter

Yang apabila pengaplikasiannya dicontohkan maka sebagai berikut

```
Learning Rate = 3 × 10⁻ ⁵

Gradient = 0.5
```

Maka parameter akan di hitung oleh sistem menjadi:

```
Parameter Baru
=
Parameter Lama
−
(3 × 10⁻ ⁵ × 0.5)
```

Proses tersebut dilakukan secara berulang pada setiap epoch hingga model mencapai performa terbaik.

- **Seqeval**

Seqeval merupakan library evaluasi yang dirancang khusus untuk tugas sequence labeling, seperti Named Entity Recognition (NER). Library ini menghitung performa model berdasarkan kecocokan label entitas antara hasil prediksi dan ground truth.

seqeval digunakan untuk menghitung nilai Precision, Recall, dan F1-Score sebagai indikator keberhasilan model IndoBERT dalam mengenali bagian teks variabel dinamis pada dokumen administratif. Proses penghitungan dari nilai yang di butuhkan sebagai berikut:

Precision	$Precision = \frac{TP}{TP + FP}$
Recall	$Recall = \frac{TP}{TP + FN}$
F1-Score	$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$

Maka sebagai contoh misal, TP = 90 , FP = 10 dan FN = 5, maka hasil yang di peroleh adalah precision = 0.90, recall = 0.947 dan F1 = 0.923

2.7.3 PDF & DOCX Processing

- **PyMuPDF**

PyMuPDF (fitz) merupakan library Python yang digunakan untuk membaca dan memanipulasi dokumen berformat PDF. Library ini mampu mengekstraksi teks, gambar, metadata, maupun struktur halaman dari dokumen PDF dengan performa yang tinggi. Selain itu, PyMuPDF juga dapat mengubah setiap halaman PDF menjadi citra (image rendering), sehingga halaman dokumen dapat diproses lebih lanjut menggunakan teknologi OCR.

Pada penelitian ini, PyMuPDF digunakan sebagai tahap awal pemrosesan dokumen PDF. Library ini membaca dokumen administratif, kemudian mengubah setiap halaman menjadi gambar agar teks pada dokumen hasil pemindaian (scanned document) dapat dikenali oleh PaddleOCR.

- **Python-DOCX**

python-docx merupakan library Python yang digunakan untuk membaca, membuat, maupun memodifikasi dokumen Microsoft Word (.docx). Library ini mampu mengakses paragraf, tabel, gambar, serta elemen-elemen lain yang terdapat pada dokumen Word.

Pada penelitian ini, python-docx digunakan untuk membaca dokumen administratif yang tersedia dalam format Microsoft Word sehingga isi dokumen dapat diekstraksi tanpa melalui proses OCR.

- **Pillow**

Pillow merupakan library pengolahan citra (image processing) yang menyediakan berbagai fungsi manipulasi gambar, seperti mengubah ukuran (resize), memotong gambar (crop), mengubah format file, serta meningkatkan kualitas citra.

Pada penelitian ini, Pillow digunakan untuk melakukan prapemrosesan gambar hasil konversi PDF sebelum diproses oleh PaddleOCR. Tahapan ini bertujuan meningkatkan kualitas gambar sehingga akurasi OCR menjadi lebih baik.

- **NumPy**

NumPy merupakan library komputasi numerik yang menyediakan struktur data array multidimensi (ndarray) serta berbagai operasi matematika yang efisien.

Pada penelitian ini, NumPy digunakan sebagai media representasi citra dalam bentuk matriks piksel sehingga dapat diproses oleh OpenCV maupun PaddleOCR.

- **OpenCV**

OpenCV (Open Source Computer Vision Library) merupakan library computer vision yang menyediakan berbagai algoritma pengolahan citra, seperti konversi warna, thresholding, filtering, deteksi tepi (edge detection), serta transformasi gambar.

Pada penelitian ini, OpenCV digunakan pada tahap prapemrosesan citra sebelum OCR, seperti mengubah gambar menjadi grayscale, melakukan thresholding, maupun memperbaiki kualitas citra sehingga karakter pada dokumen lebih mudah dikenali oleh PaddleOCR. Di library ini terjadi perhitungan untuk mengkonversi gambar menjadi grayscale, persamaannya sebagai berikut:

$$I_{gray} = 0.299R + 0.587G + 0.114B$$

Persamaan tersebut menghasilkan intensitas abu-abu berdasarkan sensitivitas mata manusia terhadap masing-masing warna.

2.7.4 OCR

- **Paddlepaddle**

PaddlePaddle merupakan framework deep learning yang dikembangkan oleh Baidu. Framework ini digunakan sebagai dasar pengembangan berbagai model kecerdasan buatan, termasuk PaddleOCR.

Pada penelitian ini, PaddlePaddle berfungsi sebagai backend yang menjalankan model OCR, mengelola proses inferensi, serta memanfaatkan GPU apabila tersedia sehingga proses pengenalan teks dapat dilakukan lebih cepat.

- **PaddleOCR**

PaddleOCR merupakan library Optical Character Recognition (OCR) yang mampu mendeteksi lokasi teks (text detection) sekaligus mengenali karakter (text recognition) pada gambar maupun dokumen hasil pemindaian.

Pada penelitian ini, PaddleOCR digunakan untuk mengekstraksi isi dokumen administratif yang berbentuk gambar atau PDF hasil scan sehingga teks dapat diproses lebih lanjut oleh model IndoBERT.

2.7.5 Utilities

- **Loguru**

Loguru merupakan library logging yang menyediakan pencatatan aktivitas (logging) aplikasi dengan konfigurasi yang lebih sederhana dibandingkan modul logging bawaan Python.

Pada penelitian ini, Loguru digunakan untuk mencatat proses OCR, ekstraksi teks, pelatihan model, maupun kesalahan (er*ror*) yang terjadi selama proses berjalan.

- **Protobuf**

Protocol Buffers (protobuf) merupakan mekanisme serialisasi data yang dikembangkan oleh Google untuk menyimpan maupun mengirim data dalam format biner yang ringkas dan efisien.

Pada penelitian ini, protobuf digunakan sebagai dependensi beberapa framework seperti PaddlePaddle sehingga pertukaran data antar komponen sistem dapat dilakukan secara efisien.

- **Pydantic**

Pydantic merupakan library validasi data berbasis type hint Python.

Pada penelitian ini, Pydantic digunakan bersama FastAPI untuk memvalidasi data masukan (request) sehingga data yang diterima API memiliki struktur dan tipe data yang sesuai.

- **Httpx**

httpx merupakan HTTP client modern yang mendukung komunikasi sinkron maupun asinkron.

Pada penelitian ini, httpx digunakan apabila sistem perlu mengakses layanan API lain atau melakukan komunikasi antar layanan (*service-to-service communication*).

- **Pandas**

pandas merupakan library analisis data yang menyediakan struktur data DataFrame sehingga memudahkan proses pembacaan, manipulasi, dan analisis dataset.

Pada penelitian ini, pandas digunakan untuk membaca dataset anotasi NER, mengelola hasil evaluasi model, serta menyimpan data eksperimen.

- **Pyarrow**

pyarrow merupakan library yang mendukung format Apache Arrow dan Parquet untuk penyimpanan data berukuran besar dengan performa tinggi.

Pada penelitian ini, pyarrow digunakan sebagai dependensi pandas ketika membaca maupun menyimpan dataset dalam format Parquet.

2.7.6 Evaluasi

- **Bertscore**

BERTScore merupakan metrik evaluasi yang mengukur kemiripan makna (semantic similarity) antara teks prediksi dan teks referensi menggunakan embedding dari model BERT.

BERTScore membandingkan representasi semantik setiap token sehingga lebih mampu mengenali sinonim maupun perubahan susunan kalimat.

Pada penelitian ini, BERTScore digunakan untuk memastikan bahwa hasil ekstraksi tidak hanya memiliki kesamaan kata, tetapi juga mempertahankan makna informasi yang terdapat pada dokumen administratif. Pada proses bertscoring ada beberapa proses penghitungan yang meliputi:

Precision	$Precision = \frac{TP}{TP + FP}$
Recall	$Recall = \frac{TP}{TP + FN}$
F1-Score	$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$

Cosine Similarity	$\text{CosSim}(x, y) = \frac{x \cdot y}{\|\|x\|\|\|\|y\|\|}$

- **Rouge Score**

rouge-score merupakan library evaluasi yang digunakan untuk menghitung tingkat kesamaan leksikal antara teks hasil prediksi dan teks referensi. Pada penelitian ini, ROUGE digunakan untuk mengevaluasi hasil ekstraksi teks dengan membandingkan isi hasil prediksi terhadap ground truth.

Library ini menyediakan tiga metrik utama, yaitu:

- ROUGE-1 (Unigram)
- ROUGE-2 (Bigram)
- ROUGE-L (Longest Common Subsequence)

2.8. Perbandingan dengan Penelitian Terdahulu

Penelitian oleh Al Faraby dkk. (2020) mengembangkan model NER untuk bahasa Indonesia menggunakan arsitektur BiLSTM-CNN dengan word-level embedding. Model ini berhasil mencapai performa F1 score sebesar 73,37% untuk pengenalan lima entitas umum (orang, organisasi, lokasi, kuantitas, dan waktu). Keunggulan penelitian ini terletak pada penggunaan deep learning yang mampu menangkap konteks sekuensial dan fitur lokal teks secara bersamaan. Namun, penelitian ini tidak mempertimbangkan aspek tata letak dokumen (layout) dan hanya berfokus pada teks biasa, sehingga kurang applicable untuk dokumen administratif yang memiliki struktur spasial penting. Selain itu, entitas yang diekstrak bersifat umum dan tidak mencakup entitas khusus administratif seperti nomor surat, jenis dokumen, atau perihal. Penelitian tahun 2025 yang memodelkan NER untuk mendeteksi entitas peristiwa pada berita online menggunakan BiLSTM-CNN menunjukkan kinerja yang baik dengan rata-rata F1-score sebesar 81%.

Penelitian terbaru oleh Zain dkk. (2025) menunjukkan bahwa augmentasi data menggunakan LLM seperti GPT-4 mampu meningkatkan kinerja model NER untuk Bahasa Indonesia melalui strategi parafrase kalimat dan substitusi entitas. Di bidang peringkasan teks, Yuliska dan Syaliman (2020) melakukan literatur review komprehensif terhadap metode, aplikasi, dan dataset peringkasan dokumen teks otomatis untuk teks berbahasa Indonesia. Review ini mencakup pendekatan unsupervised (seperti TextRank, TF-IDF, Maximum Marginal Relevance) maupun supervised (seperti sequence-to-sequence, transformer-based models), serta dataset yang tersedia seperti INDOSUM dan Liputan6. Penelitian ini menjadi fondasi penting dalam memahami landscape peringkasan teks bahasa Indonesia, namun tidak mengimplementasikan sistem secara langsung dan tidak mengintegrasikannya dengan ekstraksi informasi. Penelitian tahun 2025 oleh Winarko dkk. menunjukkan bahwa pendekatan stacked embeddings dengan transformer decoder dapat menghasilkan ringkasan teks abstraktif yang baik untuk bahasa Indonesia.

Sementara itu, penelitian di tingkat internasional oleh BinMakhashen dan Mahmoud (2020) berfokus pada analisis tata letak dokumen historis menggunakan pendekatan learning-free hybrid. Metode yang diusulkan menggabungkan analisis whitespace top-down dengan anisotropic diffusion filtering untuk menemukan area kosong antar blok teks, serta ekstraksi fitur geometris yang merepresentasikan perilaku penulisan penulis manuskrip. Dengan tingkat keberhasilan segmentasi mencapai 98,5%, penelitian ini membuktikan bahwa pendekatan berbasis aturan spasial masih sangat efektif untuk deteksi struktur dokumen, terutama ketika data latih terbatas. Namun, penelitian ini tidak melakukan analisis semantik terhadap teks yang telah tersegmentasi. Perkembangan terkini dalam document layout analysis menggunakan deep learning, seperti pendekatan berbasis YOLOv4 dan YOLOv8, mampu mengidentifikasi posisi judul, paragraf teks, tabel, dan gambar dalam dokumen dengan akurasi yang tinggi.

Posisi penelitian ini berbeda secara signifikan dari penelitian-penelitian terdahulu. Pertama, penelitian ini mengintegrasikan tiga aspek penting dalam satu pipeline: (1) analisis tata letak dokumen untuk identifikasi bagian struktural

(Header, Body, Table, Signature) menggunakan pendekatan heuristik spasial yang terinspirasi dari BinMakhashen dan Mahmoud, (2) ekstraksi entitas administratif spesifik yang lebih kaya dibandingkan penelitian Al Faraby dkk., dengan menggabungkan rule-based NER untuk entitas berformat tetap dan model machine learning untuk entitas kontekstual, serta (3) peringkasan teks otomatis menggunakan TextRank pada bagian Body, yang merujuk pada kajian Yuliska dan Syaliman. Penelitian oleh Hidayat dkk. (2025) tentang peringkasan teks ekstraktif menggunakan IndoBERT dan CNN 1D juga menunjukkan hasil yang baik untuk bahasa Indonesia.

Kedua, dari sisi dataset, penelitian ini menggunakan dokumen administratif nyata dari lingkungan JTIK, bukan teks berita atau Wikipedia seperti penelitian sebelumnya. Hal ini memungkinkan pengembangan sistem yang lebih aplikatif untuk kebutuhan institusi pendidikan. Ketiga, penelitian ini memanfaatkan informasi spasial (bounding box, ukuran font) dari PyMuPDF yang tidak tersedia dalam penelitian NER konvensional, sehingga memungkinkan deteksi bagian struktural berdasarkan posisi, bukan hanya konten teks.

BAB III
METODOLOGI PENELITIAN

3.1. Rancangan Penelitian

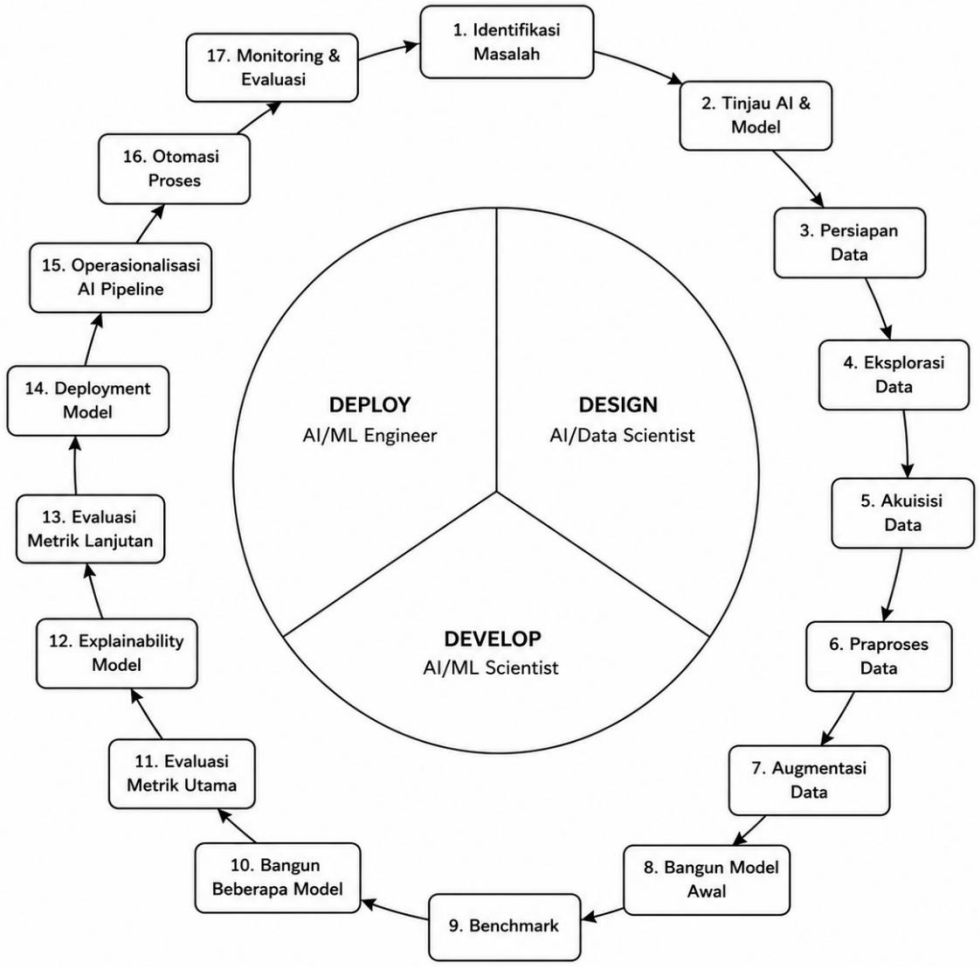
Penelitian ini menerapkan metode eksperimen yang berfokus pada rekayasa kecerdasan buatan (Artificial Intelligence Engineering), khususnya pada analisis dokumen administratif menggunakan teknologi Optical Character Recognition (OCR), Named Entity Recognition (NER), dan document summarization dalam lingkup Natural Language Processing (NLP). Pendekatan ini dipilih untuk mengembangkan, menguji, dan mengoptimalkan kinerja model deep learning dalam mengidentifikasi informasi penting serta menghasilkan ringkasan dokumen administratif secara otomatis. Fokus utama penelitian terletak pada pembangunan pipeline analisis dokumen yang terdiri dari proses ekstraksi teks menggunakan PaddleOCR, prapemrosesan data, identifikasi entitas menggunakan model IndoBERT yang telah melalui proses fine-tuning, serta pembentukan ringkasan dokumen berdasarkan informasi yang berhasil diekstraksi.

Berbeda dengan pengembangan perangkat lunak konvensional, penelitian ini menitikberatkan pada pengolahan data dokumen, pelatihan model kecerdasan buatan, dan evaluasi performa model menggunakan metrik Precision, Recall, dan F1-Score. Selain itu, dilakukan serangkaian eksperimen pelatihan dengan berbagai konfigurasi hyperparameter dan strategi optimasi untuk memperoleh model terbaik dalam mengenali entitas administratif seperti nomor surat, jenis dokumen, tanggal, pengirim, penerima, perihal, lokasi, waktu, dan isi dokumen. Pendekatan ini diharapkan mampu menghasilkan sistem analisis dokumen administratif yang akurat, efisien, dan dapat digunakan untuk membantu proses pengolahan informasi pada berbagai jenis dokumen administratif secara otomatis.

3.2. Tahapan Penelitian

Tahapan penelitian dijalankan secara sistematis mengikuti siklus rekayasa perangkat lunak, mulai dari analisis kebutuhan, perancangan arsitektur, implementasi modul, integrasi, hingga pengujian dan evaluasi. Alur kerja teknis

sistem digambarkan dalam diagram blok berikut untuk memvisualisasikan hubungan antar komponen teknologi dan alur data dari input hingga output.



Gambar 3.1 Diagram AI engineering Lifecycle

Penelitian ini menerapkan metode Artificial Intelligence Engineering Lifecycle yang terdiri atas enam tahapan utama, yaitu analisis kebutuhan, akuisisi dan pemrosesan data, pemodelan algoritma, evaluasi, implementasi sistem, serta pemeliharaan dan pemantauan. Metode ini dipilih karena sesuai dengan karakteristik penelitian yang berfokus pada pengembangan sistem kecerdasan buatan untuk melakukan ekstraksi informasi dan peringkasan dokumen administratif secara otomatis menggunakan teknologi Optical Character

Recognition (OCR), Natural Language Processing (NLP), dan Named Entity Recognition (NER).

Tahap pertama adalah analisis kebutuhan. Pada tahap ini dilakukan identifikasi permasalahan melalui studi literatur, observasi dokumen administratif yang digunakan di lingkungan TIK PNJ, serta analisis kebutuhan pengguna. Hasil dari tahap ini berupa spesifikasi kebutuhan sistem yang mencakup kebutuhan fungsional, seperti kemampuan ekstraksi informasi penting dari dokumen administratif dan pembuatan ringkasan otomatis, serta kebutuhan nonfungsional yang berkaitan dengan performa, kemudahan penggunaan, dan kompatibilitas sistem terhadap berbagai format dokumen.

Tahap kedua adalah akuisisi dan pemrosesan data. Dataset yang digunakan dalam penelitian terdiri atas 246 dokumen administratif TIK PNJ yang mencakup berbagai jenis dokumen, seperti Nota Dinas, Surat Keterangan, Surat Keterangan Lulus, Surat Permohonan, Surat Undangan, Pengumuman, dan Pengalihan Perkuliahan. Seluruh dokumen kemudian dianotasi secara manual menggunakan Label Studio untuk menghasilkan data pelatihan NER. Proses anotasi selanjutnya divalidasi bersama admin jurusan guna memastikan kesesuaian label dengan konteks administratif yang sebenarnya. Setelah proses anotasi selesai, dilakukan tahap pra-pemrosesan yang meliputi ekstraksi teks menggunakan PaddleOCR untuk dokumen hasil pemindaian, tokenisasi, normalisasi teks, serta pembentukan label menggunakan skema BIO (Begin, Inside, Outside).

Tahap ketiga adalah pemodelan algoritma. Pada tahap ini dilakukan pembangunan model Named Entity Recognition menggunakan IndoBERT yang telah di-fine-tuning pada dataset hasil anotasi. Model digunakan untuk mengenali entitas penting pada dokumen administratif seperti nomor surat, jenis dokumen, tanggal, pengirim, penerima, perihal, lokasi, isi dokumen, dan informasi tabel. Selain itu, dilakukan integrasi dengan pustaka spaCy untuk mendukung proses pemrosesan bahasa alami dan pembentukan ringkasan dokumen. Beberapa konfigurasi pelatihan dan eksperimen dilakukan untuk memperoleh model dengan performa terbaik sebelum diterapkan pada sistem.

Tahap keempat adalah evaluasi. Evaluasi dilakukan terhadap hasil pelatihan model menggunakan metrik Precision, Recall, dan F1-Score. Selain itu, digunakan confusion matrix untuk menganalisis performa model pada setiap kategori entitas yang dikenali. Pengujian juga dilakukan melalui User Acceptance Testing (UAT) untuk mengetahui tingkat penerimaan pengguna terhadap sistem yang dikembangkan. Hasil evaluasi digunakan sebagai dasar dalam menentukan model terbaik yang akan diimplementasikan pada sistem.

Tahap kelima adalah implementasi sistem. Model terbaik hasil evaluasi diintegrasikan ke dalam sistem analisis dokumen berbasis web menggunakan REST API. Sistem menerima dokumen dalam format PDF maupun DOCX, melakukan ekstraksi teks, menjalankan proses NER, menghasilkan ringkasan dokumen, serta menampilkan hasil ekstraksi informasi kepada pengguna. Tahap ini menghasilkan prototipe sistem yang dapat digunakan secara langsung untuk membantu proses analisis dokumen administratif.

Tahap terakhir adalah pemeliharaan dan pemantauan. Tahap ini bertujuan untuk memastikan sistem tetap berjalan dengan baik setelah diimplementasikan. Aktivitas yang dilakukan meliputi pemantauan performa sistem, evaluasi kualitas model secara berkala, penambahan dataset baru apabila diperlukan, serta retraining model untuk meningkatkan kemampuan sistem dalam mengenali variasi dokumen administratif yang baru. Hasil dari tahap ini berupa sistem yang dapat terus dikembangkan dan disesuaikan dengan kebutuhan pengguna di masa mendatang.

3.3. Objek Penelitian

Objek penelitian pada studi ini adalah sebuah sistem analisis dokumen administratif berbasis kecerdasan buatan yang dirancang untuk mengotomatisasi proses ekstraksi informasi penting dan pembentukan ringkasan dokumen. Sistem menerima dokumen administratif berbahasa Indonesia dalam format PDF maupun DOCX sebagai masukan utama. Dokumen yang diproses mencakup berbagai jenis dokumen administratif seperti Nota Dinas, Surat Keterangan, Surat Keterangan

Lulus, Surat Permohonan, Surat Undangan, Pengumuman, dan Pengalihan Perkuliahan. Dokumen-dokumen tersebut mengandung informasi penting yang perlu diidentifikasi secara otomatis, seperti nomor surat, jenis dokumen, tanggal, pengirim, penerima, perihal, lokasi, waktu, dan isi dokumen.

Pemrosesan dokumen dalam sistem dilakukan melalui sebuah pipeline yang terstruktur. Tahap pertama adalah ekstraksi teks dari dokumen. Untuk dokumen PDF digital digunakan metode ekstraksi teks berbasis PyMuPDF, sedangkan untuk dokumen hasil pemindaian (scanned document) digunakan teknologi Optical Character Recognition (OCR) menggunakan PaddleOCR. Untuk dokumen berformat DOCX, teks diekstraksi menggunakan pustaka python-docx. Hasil ekstraksi kemudian melalui tahap preprocessing yang meliputi normalisasi teks, pembersihan karakter yang tidak diperlukan, serta penyesuaian format agar sesuai dengan kebutuhan model pembelajaran mesin.

Teks yang telah diproses selanjutnya dianalisis menggunakan pendekatan Natural Language Processing (NLP) dengan memanfaatkan model IndoBERT yang telah melalui proses fine-tuning untuk tugas Named Entity Recognition (NER). Model ini bertugas mengidentifikasi dan mengklasifikasikan entitas penting pada dokumen administratif ke dalam kategori yang telah ditentukan, yaitu NOMOR_SURAT, JENIS_DOKUMEN, TANGGAL, PENGIRIM, PENERIMA, PERIHAL, LOKASI, WAKTU, dan ISI. Selain itu, sistem juga memanfaatkan model LayoutLMv3 untuk mendukung proses ekstraksi informasi pada dokumen yang memiliki struktur tabel atau tata letak kompleks.

Informasi yang berhasil diidentifikasi oleh model NER kemudian diintegrasikan untuk membentuk representasi terstruktur dokumen dalam format JSON. Berdasarkan hasil identifikasi tersebut, sistem menghasilkan ringkasan dokumen secara otomatis yang berisi informasi utama dan konteks dokumen secara singkat. Sistem diimplementasikan dalam bentuk REST API menggunakan framework FastAPI dan di-deploy pada layanan AWS EC2 sehingga dapat diakses oleh berbagai aplikasi atau pengguna. Sistem ini ditujukan untuk membantu proses analisis dokumen administratif secara lebih cepat, akurat, dan efisien, baik bagi

staf administrasi, pengelola dokumen, maupun pengguna umum yang membutuhkan informasi penting dari dokumen dalam waktu singkat.

3.4. Sumber Data

Populasi dalam penelitian ini adalah seluruh dokumen yang terdapat dalam arsip Jurnal Teknologi Informasi dan Komunikasi (JTIK), dengan sampel penelitian yang diambil secara purposif sebanyak 246 dari tahun 2024,2025,2026 yang telah dilabelkan. Pemilihan sampel ini mempertimbangkan keragaman tata letak dokumen ilmiah, seperti variasi penempatan kolom, tabel, dan gambar, guna menguji ketangguhan (robustness) sistem terhadap struktur dokumen yang kompleks. Instrumen penelitian utama berupa prototipe sistem perangkat lunak yang akan dijalankan melalui skrip eksperimen terstruktur, di mana dataset JTIK berperan ganda sebagai sumber data masukan sekaligus sebagai standar acuan (ground truth). Labeling atau anotasi yang telah tersedia pada 246 dokumen tersebut digunakan sebagai patokan (benchmark) untuk memvalidasi akurasi hasil ekstraksi teks spasial yang dihasilkan oleh sistem.

3.5. Teknik Pengumpulan Data

Teknik pengumpulan data yang utama adalah eksperimen dan observasi sistematis terhadap kinerja model serta sistem analisis dokumen administratif yang dikembangkan. Data dikumpulkan secara otomatis melalui proses pelatihan, validasi, dan pengujian model menggunakan kumpulan dokumen administratif yang telah dianotasi. Untuk setiap dokumen yang diproses, sistem akan mencatat dan menyimpan hasil dari setiap tahapan pipeline, seperti hasil ekstraksi teks dari PaddleOCR atau modul ekstraksi dokumen, hasil preprocessing teks, daftar entitas yang dikenali oleh model IndoBERT-NER, serta ringkasan dokumen yang dihasilkan sistem. Selain itu, data hasil pelatihan berupa training loss, validation loss, precision, recall, dan F1-score juga direkam untuk setiap skenario eksperimen yang dilakukan.

Selain data hasil prediksi, sistem juga mengumpulkan data evaluasi berupa confusion matrix yang digunakan untuk menganalisis tingkat keberhasilan identifikasi setiap kategori entitas pada dokumen administratif. Data ini digunakan

untuk mengetahui pola kesalahan klasifikasi yang masih terjadi serta mengevaluasi performa model pada masing-masing label. Sebagai pelengkap, dilakukan observasi melalui User Acceptance Testing (UAT) dengan melibatkan pengguna untuk menilai kesesuaian hasil analisis dokumen dan ringkasan yang dihasilkan sistem. Seluruh data yang diperoleh kemudian digunakan sebagai dasar dalam analisis performa model dan evaluasi efektivitas sistem secara keseluruhan.

3.6. Teknik Analisis Data

Analisis data pada penelitian ini dilakukan menggunakan pendekatan campuran (mixed-methods) yang menggabungkan analisis DATA EKSPERIMEN. Analisis kuantitatif digunakan untuk mengevaluasi performa model Named Entity Recognition (NER) menggunakan metrik Precision, Recall, dan F1-Score, serta confusion matrix untuk mengetahui tingkat keberhasilan identifikasi setiap kategori entitas pada dokumen administratif. Selain itu, hasil dari 13 skenario pelatihan dibandingkan untuk menentukan konfigurasi model terbaik yang digunakan pada sistem. Analisis kualitatif dilakukan melalui User Acceptance Testing (UAT) dan pemeriksaan manual terhadap hasil identifikasi entitas serta ringkasan dokumen untuk menilai kesesuaian keluaran sistem dengan kebutuhan pengguna. Pendekatan ini digunakan untuk memastikan bahwa sistem tidak hanya memiliki performa model yang baik secara statistik, tetapi juga dapat digunakan secara efektif dalam proses analisis dokumen administratif.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk mengidentifikasi spesifikasi sistem yang harus dipenuhi agar dapat mengatasi permasalahan pengelolaan dokumen yang telah diidentifikasi pada fase awal penelitian. Proses analisis melibatkan pengumpulan data melalui wawancara dengan pengguna potensial. Data yang terkumpul kemudian dianalisis menggunakan metode analisis konten kualitatif untuk merumuskan kebutuhan fungsional dan non-fungsional sistem.

4.1.1 Hasil Wawancara dengan Pengguna

Penggalian kebutuhan pengguna dilakukan melalui wawancara semi-terstruktur dengan tadmin jurusan dari JTIK. Wawancara dirancang untuk menggali informasi mengenai jenis dokumen yang secara rutin dikelola dalam konteks akademik, alur kerja yang diterapkan dalam proses pengelolaan dokumen tersebut, permasalahan yang muncul dari alur kerja yang berjalan, serta kebutuhan terhadap sistem yang dapat menjawab permasalahan tersebut secara efektif. Fokus penggalian kebutuhan tidak diarahkan pada satu jenis dokumen administratif tertentu, melainkan pada dokumen akademik dengan struktur yang relatif berulang dan dikelola secara langsung oleh mahasiswa. Hasil wawancara disajikan per responden berikut ini.

Jenis Dokumen yang Dikelola

Kegiatan akademik pada program studi Psikologi banyak melibatkan praktikum, observasi lapangan, dan penugasan kelompok berbasis studi kasus, sehingga sebagian besar dokumen yang dihasilkan merupakan produk kolaboratif. Jenis dokumen yang paling sering dikelola mencakup proposal mini praktikum, rancangan observasi, laporan hasil diskusi kelompok, dan laporan akhir praktikum. Dokumen-dokumen tersebut memiliki struktur yang relatif seragam dengan bagian-bagian seperti pendahuluan, tujuan kegiatan, metode atau alur pelaksanaan, pembagian peran anggota, hasil pengamatan, dan penutup.

Format dasar dokumen umumnya telah ditentukan oleh dosen pengampu atau disepakati di awal semester, sehingga kerangka struktural jarang mengalami perubahan meskipun topik dan isi kegiatan berbeda dari satu pertemuan ke pertemuan berikutnya. Konsistensi format ini menjadikan dokumen-dokumen tersebut sebagai kandidat yang sesuai untuk dikelola melalui sistem berbasis template.

Alur Kerja Pengelolaan Dokumen

Dalam praktik pengelolaan dokumen kelompok, dokumen dari tugas sebelumnya digunakan sebagai acuan struktural. Dokumen lama dibuka kembali untuk melihat susunan penulisan, kemudian disesuaikan secara manual untuk kebutuhan tugas baru. Proses kolaborasi dilakukan dengan cara saling mengirim file antara anggota kelompok atau memanfaatkan layanan penyimpanan cloud secara terpisah, sehingga masing-masing anggota berpotensi memiliki versi dokumen yang berbeda pada waktu yang bersamaan.

Permasalahan yang Dihadapi

Penggunaan alur kerja berbasis file transfer mengakibatkan sejumlah permasalahan yang berulang. Kesulitan membedakan versi dokumen yang hampir serupa menjadi kendala utama, diperparah oleh banyaknya file dengan nama yang mirip namun konteks kegiatannya berbeda. Dokumen yang tersebar di berbagai folder dan akun penyimpanan milik masing-masing anggota mempersulit koordinasi dan pelacakan dokumen lama yang diperlukan sebagai referensi struktur. Kondisi ini menyebabkan inefisiensi dalam proses pengelolaan dokumen, terutama ketika dokumen dari satu mata kuliah perlu diadaptasi untuk digunakan pada konteks mata kuliah lain.

Kebutuhan terhadap Sistem

Kebutuhan yang teridentifikasi dari wawancara ini meliputi pengelolaan dokumen kelompok yang lebih terstruktur, pengelompokan dokumen berdasarkan mata kuliah atau jenis kegiatan, pemeliharaan konsistensi struktur dokumen, serta dukungan kolaborasi yang tidak menimbulkan konflik versi. Kebutuhan ini secara

spesifik mengarah pada pentingnya fitur kolaborasi real-time dan manajemen versi sebagai komponen kunci dalam sistem yang akan dikembangkan.

4.1.2 Identifikasi Kebutuhan Fungsional

Kebutuhan fungsional merupakan sekumpulan fungsi dan layanan yang harus disediakan oleh sistem agar tujuan penelitian dapat tercapai. Pada penelitian ini, sistem dikembangkan untuk melakukan identifikasi bagian teks variabel dinamis pada dokumen administratif secara otomatis menggunakan kombinasi teknologi Optical Character Recognition (OCR), Natural Language Processing (NLP), dan Named Entity Recognition (NER). Oleh karena itu, kebutuhan fungsional sistem dirumuskan berdasarkan alur kerja pemrosesan dokumen mulai dari tahap penerimaan dokumen hingga menghasilkan informasi terstruktur dan ringkasan dokumen.

Proses analisis kebutuhan dilakukan dengan mengidentifikasi aktivitas utama yang harus dilakukan oleh sistem ketika menerima dokumen administratif dari pengguna. Aktivitas tersebut meliputi penerimaan dokumen dalam berbagai format, ekstraksi teks dari dokumen digital maupun hasil pemindaian, pembersihan data teks, identifikasi entitas penting, deteksi informasi tabel, hingga pembentukan ringkasan dokumen secara otomatis. Setiap aktivitas tersebut kemudian diterjemahkan menjadi fungsi-fungsi yang harus tersedia pada sistem sehingga seluruh proses analisis dapat berjalan secara terintegrasi.

Selain fungsi yang berkaitan langsung dengan proses ekstraksi informasi, sistem juga dirancang untuk mendukung kebutuhan integrasi dengan aplikasi lain melalui layanan berbasis REST API. Pendekatan ini dipilih karena memungkinkan sistem digunakan sebagai layanan analisis dokumen yang dapat diakses dari berbagai platform tanpa bergantung pada implementasi antarmuka tertentu. Dengan adanya API, sistem dapat dimanfaatkan tidak hanya sebagai aplikasi mandiri, tetapi juga sebagai komponen pendukung pada sistem administrasi, manajemen dokumen, maupun layanan digital lainnya.

Kebutuhan fungsional yang telah diidentifikasi selanjutnya digunakan sebagai dasar dalam proses perancangan dan implementasi sistem. Setiap kebutuhan akan direalisasikan melalui modul tertentu pada pipeline pemrosesan dokumen

sehingga seluruh fitur yang dibangun memiliki keterkaitan langsung dengan tujuan penelitian.

Tabel 4.1 Tabel Kebutuhan Functional

No.	Kebutuhan Fungsional	Deskripsi	Prioritas
KF-01	Unggah Dokumen	Sistem mampu menerima dokumen administratif dalam format PDF dan DOCX sebagai input analisis.	Tinggi
KF-02	Deteksi Jenis Dokumen	Sistem mampu membedakan dokumen PDF digital dan PDF hasil pemindaian.	Tinggi
KF-03	Ekstraksi Teks Dokumen	Sistem mampu mengekstraksi teks dari dokumen menggunakan PyMuPDF dan PaddleOCR.	Tinggi
KF-04	Preprocessing Teks	Sistem mampu membersihkan teks dengan menghapus header, footer, dan kop surat.	Tinggi
KF-05	Identifikasi Entitas	Sistem mampu mengenali entitas administratif menggunakan model NER.	Tinggi
KF-06	Deteksi Informasi Tabel	Sistem mampu mengekstraksi informasi tabel menggunakan LayoutLMv3.	Menengah
KF-07	Post-Processing Entitas	Sistem mampu melakukan normalisasi dan validasi hasil ekstraksi entitas.	Tinggi

No.	Kebutuhan Fungsional	Deskripsi	Prioritas
KF-08	Ekspor Dokumen	Unduhan dokumen dalam format DOCX dan PDF	Tinggi
KF-09	REST API Service	Sistem menyediakan endpoint API untuk integrasi dengan aplikasi eksternal.	Tinggi

4.1.3 Identifikasi Kebutuhan Non-Fungsional

Dalam penelitian ini, kebutuhan nonfungsional dirumuskan berdasarkan karakteristik model Artificial Intelligence yang digunakan serta lingkungan implementasi sistem. Sistem memanfaatkan model berbasis Transformer, OCR, dan document understanding yang memerlukan sumber daya komputasi relatif besar dibandingkan aplikasi konvensional. Oleh karena itu, aspek performa, efisiensi penggunaan memori, serta kemampuan skalabilitas menjadi faktor penting yang harus dipertimbangkan selama proses pengembangan sistem.

Selain aspek performa, kualitas hasil ekstraksi informasi juga menjadi perhatian utama. Sistem harus mampu menghasilkan keluaran yang konsisten dan akurat pada berbagai jenis dokumen administratif. Hal ini penting karena hasil ekstraksi akan digunakan sebagai dasar dalam proses pembentukan ringkasan maupun pengambilan informasi penting dari dokumen. Dengan demikian, tingkat akurasi model menjadi salah satu kebutuhan nonfungsional yang memiliki prioritas tinggi dalam penelitian ini.

Sistem juga dirancang untuk diimplementasikan pada lingkungan cloud menggunakan layanan AWS EC2. Oleh karena itu, kebutuhan terkait ketersediaan layanan, kemudahan integrasi, dan maintainability menjadi faktor yang harus dipenuhi. Struktur aplikasi yang modular memungkinkan pengembangan lebih lanjut dilakukan tanpa harus mengubah keseluruhan sistem. Selain itu, penggunaan FastAPI memberikan kemudahan dalam penyediaan dokumentasi

API secara otomatis sehingga proses integrasi dengan aplikasi lain dapat dilakukan dengan lebih mudah.

Tabel 4.2 Tabel Kebutuhan non functional

No.	Aspek	Deskripsi Persyaratan
KNF-01	Performa Pemrosesan	Sistem mampu memproses dokumen secara efisien sesuai kapasitas server.
KNF-02	Akurasi Ekstraksi	Sistem mampu menghasilkan identifikasi entitas dengan tingkat akurasi yang tinggi.
KNF-03	Skalabilitas	Sistem mampu menangani peningkatan jumlah permintaan pengguna.
KNF-04	Ketersediaan Layanan	Sistem dapat diakses selama layanan cloud aktif.
KNF-05	Kompatibilitas Format	Sistem mampu memproses PDF digital, PDF scan, dan DOCX.
KNF-06	Kemudahan Integrasi	Sistem menyediakan REST API yang mudah diintegrasikan dengan aplikasi lain.
KNF-07	Efisiensi Memori	Sistem menggunakan lazy loading untuk mengurangi penggunaan memori server.
KNF-08	Maintainability	Sistem memiliki struktur modular yang mudah dikembangkan dan dipelihara.
KNF-09	Reliability	Sistem menghasilkan keluaran yang konsisten pada berbagai jenis dokumen.

4.1.4 Prioritas Kebutuhan

Penentuan prioritas kebutuhan dilakukan untuk mengidentifikasi tingkat kepentingan masing-masing kebutuhan terhadap tujuan penelitian. Prioritas digunakan sebagai acuan dalam proses perancangan dan implementasi sistem sehingga pengembangan dapat difokuskan terlebih dahulu pada fitur-fitur yang memiliki pengaruh terbesar terhadap keberhasilan sistem. Pada penelitian ini, prioritas kebutuhan dibagi menjadi tiga kategori yaitu prioritas tinggi, prioritas sedang, dan prioritas rendah.

Kebutuhan dengan prioritas tinggi merupakan kebutuhan inti yang secara langsung mendukung proses identifikasi bagian teks variabel dinamis pada dokumen administratif. Kebutuhan ini harus tersedia agar sistem dapat menjalankan fungsi utamanya sesuai dengan tujuan penelitian. Kebutuhan dengan prioritas sedang berfungsi sebagai fitur pendukung yang meningkatkan kualitas dan fleksibilitas penggunaan sistem. Sementara itu, kebutuhan dengan prioritas rendah merupakan kebutuhan tambahan yang tidak secara langsung memengaruhi proses utama namun dapat meningkatkan kemudahan operasional dan pemeliharaan sistem.

Tabel 4.3 Matriks Prioritas Kebutuhan

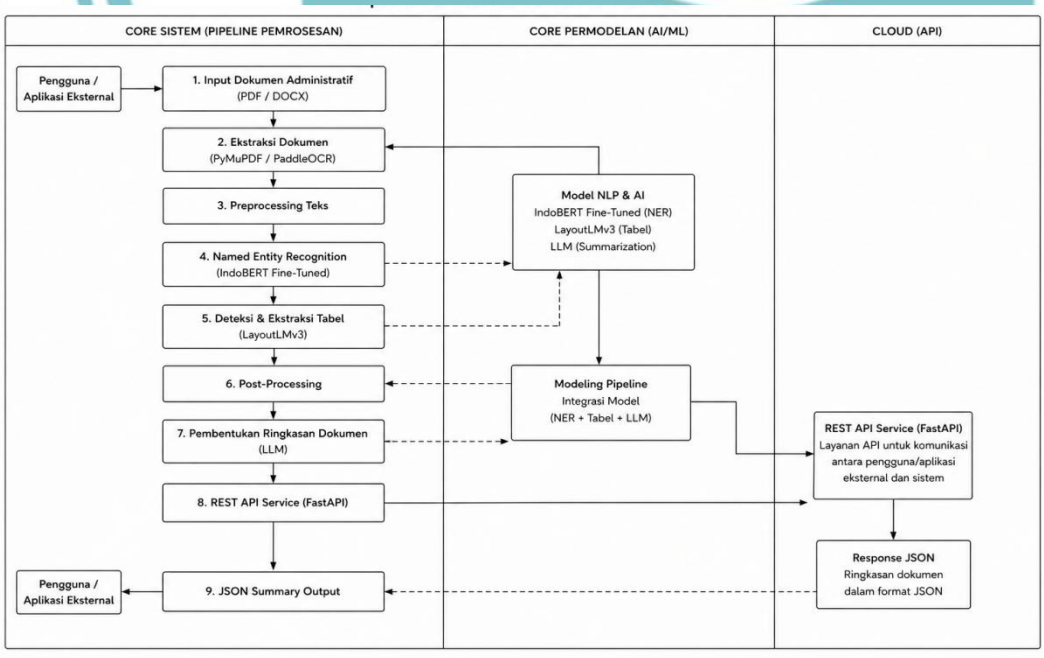
Prioritas	Kebutuhan	Justifikasi
Tinggi	KF-01, KF-02, KF-03, KF-04, KF-05, KF-07, KF-08,	Muncul konsisten pada seluruh responden; kritis untuk fungsi inti sistem
Menengah	KF-06, KF-09, KF	Kontekstual; meningkatkan efektivitas namun tidak menghambat fungsi dasar
Rendah	Pembuatan Template dan Penyuntingan secara Online	Nilai tambah; tidak berkaitan langsung dengan permasalahan inti

4.2 Perancangan Sistem

Sistem dirancang menggunakan pendekatan modular yang terdiri dari beberapa komponen utama yaitu modul ekstraksi dokumen, modul preprocessing, modul Named Entity Recognition, modul deteksi tabel, modul post-processing, modul pembangkitan ringkasan, dan modul API service. Setiap komponen memiliki tugas spesifik yang saling terhubung sehingga membentuk sebuah pipeline pemrosesan dokumen yang terintegrasi.

4.2.1 Arsitektur Sistem

Perancangan arsitektur sistem dilakukan untuk menggambarkan hubungan antar komponen yang terlibat dalam proses analisis dokumen administratif. Arsitektur ini dirancang menggunakan pendekatan modular agar setiap komponen dapat bekerja secara independen namun tetap terintegrasi dalam satu alur pemrosesan. Pendekatan tersebut memungkinkan sistem untuk lebih mudah dikembangkan, dipelihara, dan diintegrasikan dengan layanan lain di masa mendatang. Selain itu, penggunaan arsitektur modular juga membantu dalam pemisahan tanggung jawab setiap komponen sehingga proses pengembangan dapat dilakukan secara lebih terstruktur.



Gambar 4.1 Diagram Arsitektur Sistem

Pada Diagram arsitektur sistem, sistem dibagi ke dalam tiga lapisan utama, yaitu Core Sistem yang menangani alur pemrosesan dokumen, Core Permodelan yang mengelola model kecerdasan buatan, serta Cloud API yang berfungsi sebagai antarmuka komunikasi dengan pengguna atau aplikasi eksternal.

Proses dimulai ketika pengguna atau aplikasi eksternal mengirimkan dokumen administratif dalam format PDF maupun DOCX ke dalam sistem. Dokumen yang diterima kemudian memasuki tahap ekstraksi dokumen. Pada tahap ini, sistem menentukan metode ekstraksi yang sesuai berdasarkan karakteristik dokumen. Dokumen digital diproses menggunakan PyMuPDF untuk memperoleh teks secara langsung, sedangkan dokumen hasil pemindaian diproses menggunakan PaddleOCR untuk mengubah informasi visual menjadi teks yang dapat dipahami oleh sistem.

Hasil ekstraksi kemudian memasuki tahap preprocessing teks. Tahap ini bertujuan membersihkan data dari berbagai elemen yang tidak relevan seperti header, footer, nomor halaman, karakter khusus, maupun noise lainnya yang dapat memengaruhi performa model. Selain itu, dilakukan normalisasi teks untuk menghasilkan data yang lebih konsisten dan siap digunakan pada proses analisis berikutnya.

Teks yang telah diproses selanjutnya digunakan sebagai masukan bagi model Named Entity Recognition (NER) berbasis IndoBERT Fine-Tuned yang berada pada lapisan Core Permodelan. Model ini bertugas mengidentifikasi berbagai entitas penting yang terdapat dalam dokumen administratif, seperti nama orang, organisasi, lokasi, tanggal, nomor dokumen, maupun informasi administratif lainnya yang relevan. Hasil identifikasi entitas tersebut digunakan untuk membantu sistem memahami konteks dokumen secara lebih mendalam.

Selain proses identifikasi entitas, sistem juga menjalankan proses deteksi dan ekstraksi tabel menggunakan model LayoutLMv3. Model ini memungkinkan sistem memahami struktur tata letak dokumen dan mengekstraksi informasi yang tersimpan dalam bentuk tabel. Dengan demikian, informasi yang tidak tersaji dalam bentuk paragraf tetap dapat dimanfaatkan dalam proses analisis dan pembentukan ringkasan.

Informasi hasil identifikasi entitas dan deteksi tabel kemudian diproses pada tahap modeling pipeline. Tahap ini berfungsi mengintegrasikan seluruh informasi yang diperoleh dari model NER, LayoutLMv3, serta data tekstual hasil ekstraksi untuk membentuk representasi dokumen yang lebih lengkap. Hasil integrasi tersebut selanjutnya dikirim kembali ke lapisan pemrosesan untuk menjalani tahap post-processing.

Pada tahap post-processing dilakukan validasi, normalisasi, dan penyempurnaan hasil ekstraksi agar data yang dihasilkan memiliki kualitas yang lebih baik dan konsisten. Setelah proses tersebut selesai, sistem membentuk ringkasan dokumen menggunakan model Large Language Model (LLM). Model ini menghasilkan ringkasan yang lebih ringkas, informatif, dan tetap mempertahankan informasi penting yang terdapat pada dokumen asli.

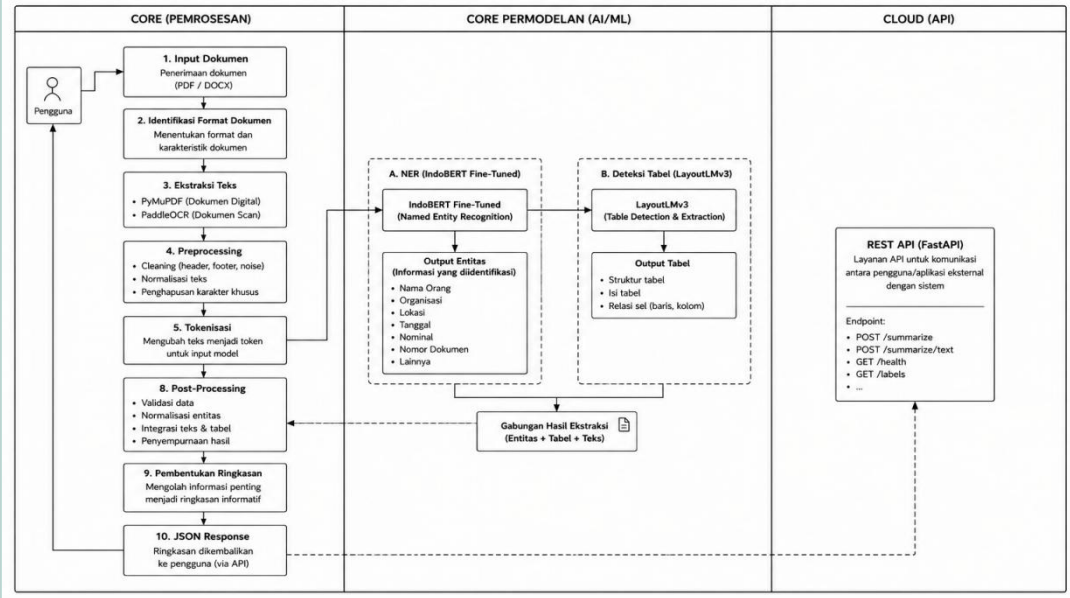
Hasil ringkasan yang telah terbentuk kemudian diteruskan ke layanan REST API yang berada pada lapisan Cloud API. Komponen ini berfungsi sebagai penghubung antara sistem dengan pengguna atau aplikasi eksternal melalui mekanisme komunikasi berbasis HTTP. REST API menerima permintaan pemrosesan dokumen, mengelola proses komunikasi dengan modul pemrosesan, serta mengirimkan hasil akhir kepada pengguna.

Tahap terakhir dari keseluruhan proses adalah pembentukan keluaran dalam format JSON. Format ini dipilih karena ringan, mudah diproses, serta mendukung integrasi dengan berbagai platform dan aplikasi eksternal. Melalui arsitektur tersebut, sistem mampu mengintegrasikan proses ekstraksi dokumen, identifikasi entitas, analisis struktur tabel, pembentukan ringkasan, dan penyajian hasil melalui layanan API dalam satu alur pemrosesan yang terstruktur dan modular.

4.2.2 Perancangan Pipeline Pemprosesan Dokumen

Pipeline pemrosesan dokumen dirancang untuk memastikan setiap tahapan analisis berjalan secara berurutan dan terstruktur. Pipeline ini menggambarkan bagaimana data bergerak dari satu proses ke proses lainnya hingga menghasilkan

keluaran akhir berupa ringkasan dokumen. Dengan adanya pipeline yang jelas, proses pengembangan dan evaluasi sistem dapat dilakukan dengan lebih mudah karena setiap tahapan memiliki fungsi dan tujuan yang spesifik.



Gambar 4.2 Diagram Pipeline System

Proses dimulai ketika pengguna mengirimkan dokumen administratif dalam format PDF atau DOCX ke dalam sistem. Dokumen yang diterima selanjutnya memasuki tahapan identifikasi format dokumen untuk menentukan karakteristik dokumen dan metode ekstraksi yang sesuai. Tahap ini penting untuk membedakan dokumen digital yang dapat diekstraksi secara langsung dengan dokumen hasil pemindaian yang memerlukan proses Optical Character Recognition (OCR).

Setelah format dokumen berhasil diidentifikasi, sistem menjalankan proses ekstraksi teks. Pada tahap ini, dokumen digital diproses menggunakan PyMuPDF untuk memperoleh informasi tekstual secara langsung, sedangkan dokumen hasil pemindaian diproses menggunakan PaddleOCR untuk mengubah informasi visual menjadi teks yang dapat dipahami oleh sistem. Hasil dari proses ini berupa teks mentah yang masih mengandung berbagai elemen non-informatif.

Teks hasil ekstraksi kemudian memasuki tahap preprocessing. Pada tahap ini dilakukan pembersihan data dengan menghilangkan noise seperti header, footer, nomor halaman, kop surat, dan karakter khusus yang tidak memiliki kontribusi

terhadap proses analisis dokumen. Selain itu, dilakukan normalisasi teks untuk meningkatkan kualitas data yang akan digunakan pada tahap pemodelan.

Setelah proses preprocessing selesai dilakukan, teks diubah menjadi representasi token melalui tahap tokenisasi. Tahap ini bertujuan menghasilkan format masukan yang sesuai dengan kebutuhan model Transformer sehingga informasi tekstual dapat diproses secara efektif oleh model pembelajaran mendalam yang digunakan dalam sistem.

Pada swimlane Core Permodelan (AI/ML), token hasil pemrosesan digunakan sebagai masukan bagi model Named Entity Recognition (NER) berbasis IndoBERT Fine-Tuned. Model ini bertugas mengidentifikasi berbagai entitas penting yang terdapat pada dokumen administratif, seperti nama orang, organisasi, lokasi, tanggal, nominal, nomor dokumen, dan informasi relevan lainnya. Informasi hasil identifikasi entitas digunakan untuk membantu sistem memahami konteks dokumen secara lebih mendalam.

Selain identifikasi entitas, sistem juga memanfaatkan model LayoutLMv3 untuk melakukan deteksi dan ekstraksi informasi tabel. Model ini mampu memahami tata letak dokumen sehingga informasi yang tersimpan dalam bentuk tabel dapat diekstraksi beserta struktur hubungan antar baris dan kolomnya. Hasil dari proses NER dan deteksi tabel kemudian digabungkan dengan informasi teks hasil ekstraksi untuk membentuk representasi dokumen yang lebih lengkap.

Seluruh hasil ekstraksi yang diperoleh dari proses pemodelan kemudian memasuki tahap post-processing pada swimlane Core Pemrosesan. Tahap ini bertujuan melakukan validasi, normalisasi, integrasi, dan penyempurnaan data sehingga informasi yang dihasilkan memiliki konsistensi dan kualitas yang baik. Setelah proses tersebut selesai, sistem membentuk ringkasan dokumen berdasarkan informasi penting yang diperoleh dari teks, entitas yang teridentifikasi, serta informasi yang berasal dari tabel.

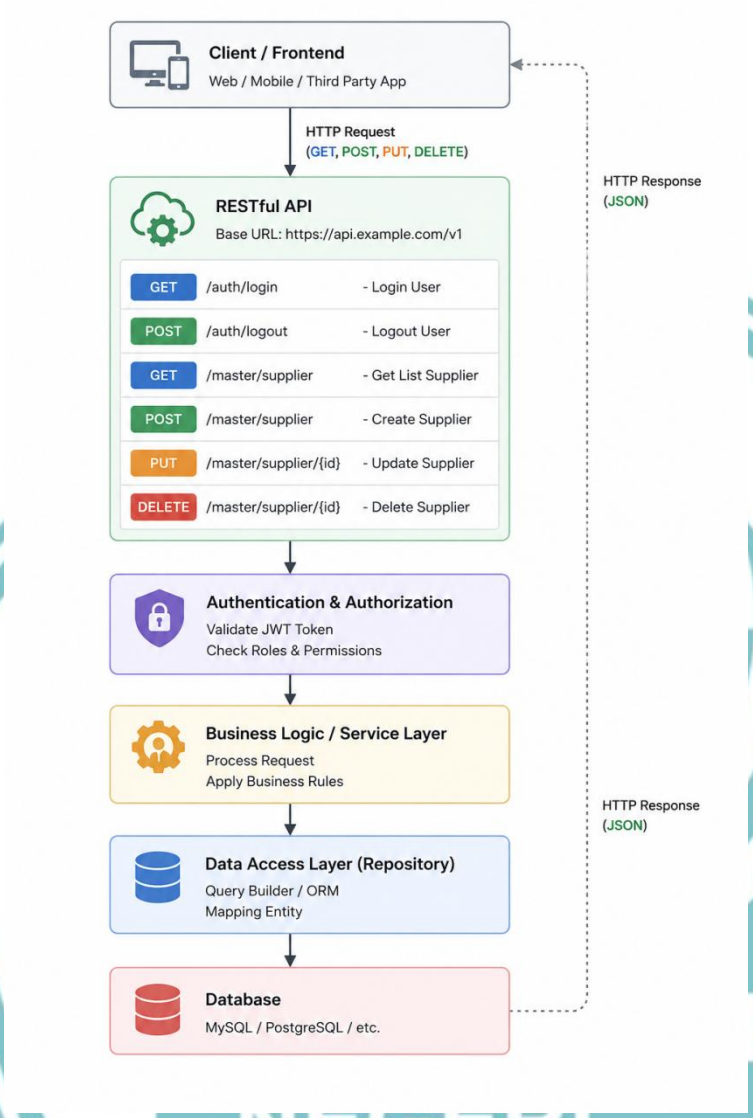
Pada swimlane Cloud (API), sistem menyediakan layanan REST API berbasis FastAPI yang berfungsi sebagai antarmuka komunikasi antara pengguna atau aplikasi eksternal dengan modul pemrosesan dokumen. Layanan ini menerima

permintaan pemrosesan dokumen, mengelola proses komunikasi dengan sistem, serta mengembalikan hasil pemrosesan melalui berbagai endpoint yang tersedia.

Hasil akhir dari keseluruhan proses berupa ringkasan dokumen yang dikembalikan kepada pengguna dalam format JSON melalui layanan API. Format JSON dipilih karena ringan, mudah diproses, serta mendukung integrasi dengan berbagai aplikasi dan layanan eksternal. Dengan pembagian komponen ke dalam swimlane Core Pemrosesan, Core Permodelan AI/ML, dan Cloud API, arsitektur yang diusulkan mampu memisahkan fungsi pemrosesan data, pemodelan kecerdasan buatan, serta layanan integrasi secara lebih terstruktur dan modular.

4.2.3 Perancangan Arsitektur API

Arsitektur REST API dirancang sebagai lapisan komunikasi antara pengguna dan sistem analisis dokumen. API berfungsi menerima permintaan dari pengguna, meneruskan permintaan tersebut ke modul pemrosesan dokumen, serta mengembalikan hasil analisis dalam format yang mudah dipahami oleh aplikasi lain. Penggunaan REST API memungkinkan sistem digunakan secara fleksibel tanpa bergantung pada platform tertentu.



Gambar 4.3 Diagram Arsitektur API

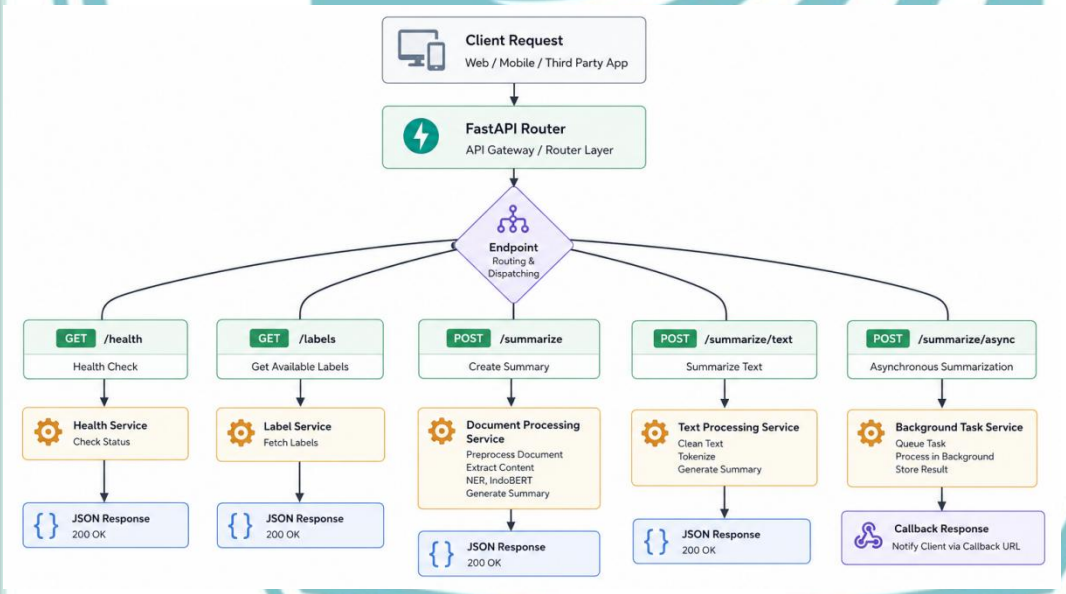
pengguna mengirimkan permintaan melalui client yang dapat berupa aplikasi web, aplikasi mobile, maupun aplikasi backend lainnya. Permintaan tersebut diterima oleh FastAPI yang bertugas melakukan validasi terhadap data masukan. Setelah validasi berhasil dilakukan, permintaan diteruskan ke layanan pemrosesan dokumen yang menjalankan seluruh pipeline analisis.

Layanan pemrosesan dokumen memanfaatkan model IndoBERT untuk melakukan identifikasi entitas dan LayoutLMv3 untuk memahami struktur tabel yang terdapat pada dokumen. Hasil ekstraksi tersebut kemudian digunakan untuk membentuk ringkasan dokumen yang akan dikembalikan kepada pengguna dalam format JSON.

Arsitektur ini memungkinkan sistem beroperasi sebagai layanan independen yang dapat diakses oleh berbagai aplikasi tanpa perlu mengetahui detail implementasi model yang digunakan. Selanjutnya, hasil perancangan yang telah dijelaskan pada bagian ini akan direalisasikan melalui proses implementasi sistem yang dibahas pada subbab berikutnya.

4.2.4 Perancangan Endpoint API

Perancangan endpoint API dilakukan untuk menyediakan mekanisme komunikasi antara pengguna dan sistem analisis dokumen administratif. Endpoint API berfungsi sebagai titik akses utama yang digunakan untuk mengirim dokumen, melakukan analisis teks, memantau status layanan, serta menerima hasil ringkasan dokumen dalam format JSON. Penggunaan REST API memungkinkan sistem diintegrasikan dengan berbagai aplikasi tanpa bergantung pada platform tertentu.



Gambar 4.4 Diagram Endpoint API

seluruh permintaan pengguna diterima oleh FastAPI Router yang bertugas mengarahkan request menuju endpoint yang sesuai. Endpoint yang tersedia terdiri atas layanan monitoring, informasi label entitas, analisis dokumen, analisis teks, dan pemrosesan asynchronous. Setiap endpoint akan meneruskan permintaan ke modul pemrosesan yang sesuai sebelum menghasilkan respons dalam format JSON.

Tabel 4.4 Tabel Daftar Endpoint API

Endpoint	Method	Function
/api/v1/health	GET	Menampilkan status layanan
/api/v1/labels	GET	Menampilkan daftar label NER
/api/v1/summarize	POST	Analisis dokumen PDF atau DOCX
/api/v1/summarize/text	POST	Analisis teks langsung
/api/v1/summarize/async	POST	Analisis dokumen secara asynchronous

Berdasarkan Tabel 4.4, endpoint /api/v1/summarize menjadi endpoint utama yang menjalankan seluruh pipeline analisis dokumen. Selain itu, sistem juga menyediakan endpoint untuk analisis teks secara langsung dan pemrosesan asynchronous menggunakan callback URL. Dengan rancangan endpoint tersebut, sistem dapat melayani berbagai kebutuhan pengguna secara fleksibel sekaligus mendukung integrasi dengan aplikasi eksternal.

4.3 Permodelan Algoritma

Tahap implementasi sistem dilakukan berdasarkan hasil perancangan yang telah disusun pada bab sebelumnya. Implementasi bertujuan untuk merealisasikan seluruh komponen yang telah dirancang menjadi sebuah sistem yang mampu melakukan analisis dokumen administratif secara otomatis. Sistem dikembangkan menggunakan bahasa pemrograman Python dengan framework FastAPI sebagai layanan API, model IndoBERT untuk Named Entity Recognition (NER),

PaddleOCR untuk ekstraksi teks dari dokumen hasil pemindaian, serta LayoutLMv3 untuk memahami struktur dokumen dan informasi tabel. Seluruh komponen diintegrasikan dalam satu pipeline pemrosesan sehingga sistem dapat menghasilkan ringkasan dokumen dalam format JSON secara otomatis.

4.3.1 Arsitektur Model

Arsitektur model yang dikembangkan pada penelitian ini menggunakan pendekatan pipeline berbasis Artificial Intelligence Engineering yang mengintegrasikan beberapa komponen pemrosesan dokumen dan Natural Language Processing dalam satu alur kerja. Tujuan utama dari arsitektur ini adalah untuk mengubah dokumen administratif yang masih berbentuk tidak terstruktur menjadi informasi terstruktur yang dapat diproses secara otomatis oleh sistem.

Pada tahap pelatihan, dataset yang terdiri atas 246 dokumen administratif TIK PNJ terlebih dahulu melalui proses validasi untuk memastikan kualitas anotasi yang telah dibuat menggunakan Label Studio. Setelah proses validasi selesai, data diperluas menggunakan teknik augmentasi untuk meningkatkan variasi data pelatihan. Dataset kemudian dibagi menjadi data pelatihan, validasi, dan pengujian sebelum diubah menjadi format yang sesuai dengan kebutuhan model IndoBERT. Model IndoBERT selanjutnya dilakukan fine-tuning menggunakan data hasil anotasi sehingga mampu mengenali entitas administratif yang spesifik seperti nomor surat, tanggal, pengirim, penerima, perihal, dan informasi penting lainnya.

Pada tahap inferensi, sistem menerima dokumen dalam format PDF atau DOCX sebagai masukan. Sistem kemudian melakukan identifikasi jenis dokumen untuk menentukan metode ekstraksi teks yang akan digunakan. Untuk dokumen PDF digital, teks diekstraksi secara langsung menggunakan PyMuPDF, sedangkan untuk dokumen hasil pemindaian digunakan PaddleOCR untuk mengubah citra dokumen menjadi teks digital. Hasil ekstraksi teks kemudian diproses menggunakan spaCy untuk normalisasi dan tokenisasi sebelum dikirim ke model IndoBERT.

Model IndoBERT yang telah dilatih bertugas melakukan Named Entity Recognition (NER) guna mengidentifikasi berbagai informasi penting yang terdapat pada dokumen. Hasil identifikasi entitas selanjutnya diproses pada tahap post-processing untuk menyusun ringkasan dokumen serta membentuk keluaran dalam format JSON. Output akhir sistem berisi metadata dokumen, hasil ekstraksi entitas, tingkat kelengkapan informasi, serta ringkasan isi dokumen yang dapat digunakan untuk mempercepat proses pencarian dan pemahaman informasi administratif.

```
def train(
    dataset_path,
    output_dir,
    augment=True,
    augment_n=5,
    epochs=20,
):

    raw_data = json.load(open(dataset_path))

    report = validate_dataset(raw_data)

    augmented_data = augment_dataset(
        raw_data,
        n_augment=augment_n
    )

    train_data, val_data, test_data = split_dataset(
        augmented_data
    )

    train_samples = build_training_samples(train_data)

    model = build_model()

    train_model(model, train_samples)

    torch.save(model.state_dict(), model_path)
```

Potongan kode tersebut menunjukkan alur utama proses pelatihan model yang digunakan dalam penelitian. Dataset yang telah dianotasi terlebih dahulu divalidasi untuk memastikan kualitas label, kemudian dilakukan augmentasi data guna meningkatkan jumlah variasi sampel pelatihan. Setelah itu dataset dibagi menjadi data pelatihan, validasi, dan pengujian. Data yang telah diproses kemudian digunakan untuk membangun sampel pelatihan dan melakukan fine-tuning model IndoBERT. Model yang telah selesai dilatih selanjutnya disimpan dan digunakan pada tahap inferensi untuk melakukan ekstraksi informasi dari dokumen administratif baru.

4.3.2 Represetasi Data dan label entitas

Dataset yang digunakan dalam penelitian ini berjumlah 2 dokumen administratif dari jurusan TIK PNJ. Dokumen-dokumen tersebut terdiri dari berbagai jenis surat resmi seperti surat keputusan, surat undangan, nota dinas, pengumuman, surat keterangan, dan surat tugas. Proses akuisisi data dilakukan dengan mengumpulkan dokumen dalam format PDF dan DOCX dari arsip jurusan. Setiap dokumen kemudian dianotasi menggunakan Label Studio, sebuah tools open-source untuk pelabelan data. Anotasi dilakukan dengan menandai entitas-entitas yang ingin dikenali langsung pada teks dokumen.

Validasi dataset dilakukan bersama admin jurusan untuk memastikan kebenaran dan konsistensi anotasi. Hal ini penting karena kualitas dataset sangat mempengaruhi performa model NER yang akan dilatih. Setiap entitas yang dianotasi diperiksa kembali oleh admin jurusan yang memahami struktur dan konten dokumen administratif.

```
# config.py:30-40
# LOCAL_LABELS mendefinisikan 9 entitas utama yang akan diekstrak
# dari dokumen surat Indonesia.
LOCAL_LABELS = [
    "NOMOR_SURAT",    # Label 1 - Nomor surat (contoh: 001/SK/2024)
    "JENIS_DOKUMEN",  # Label 2 - Jenis dokumen (contoh: SURAT
KEPUTUSAN)
    "TANGGAL",        # Label 3 - Tanggal surat (contoh: 12 Januari
2024)
    "PENGIRIM",       # Label 4 - Pengirim surat (contoh: Direktur
PNJ)
```

```
        "PENERIMA",         # Label 5 - Penerima surat (contoh: Kepala
    Divisi)
        "PERIHAL",         # Label 6 - Perihal surat (contoh: Penetapan
    Jabatan)
        "ISI",              # Label 7 - Isi utama dokumen (rule-based +
    NER)
        "TABEL",            # Label 8 - Blok tabel (rule-based hybrid +
    LayoutLMv3)
        "LOKASI",          # Label 9 - Lokasi dalam dokumen
]
```

Kode di atas merupakan daftar entitas yang menjadi target ekstraksi sistem. Setiap label merepresentasikan informasi spesifik yang umum ditemukan dalam surat resmi Indonesia.

4.3.3 Pemrosesan dan Persiapan Data

Tahap pemrosesan dan persiapan data mencakup serangkaian langkah untuk mengubah dokumen mentah (PDF/DOCX) menjadi format yang siap digunakan oleh model NER. Proses ini meliputi deteksi jenis PDF, ekstraksi teks, cropping header/footer, OCR untuk dokumen scan, pembersihan teks, pembuatan BIO tags, alignment token, dan pembagian dataset.

4.3.3.1 Deteksi Jenis PDF

Tahap awal dalam proses pemrosesan dokumen adalah melakukan identifikasi terhadap jenis PDF yang diterima oleh sistem. Proses ini bertujuan untuk menentukan metode ekstraksi teks yang paling sesuai berdasarkan karakteristik dokumen. Secara umum, dokumen PDF dapat dibedakan menjadi dua kategori, yaitu *pure PDF* yang menyimpan informasi teks secara langsung dan *scanned PDF* yang hanya berisi representasi citra hasil pemindaian. Perbedaan karakteristik tersebut menyebabkan kebutuhan pemrosesan yang berbeda, sehingga identifikasi jenis dokumen menjadi langkah penting sebelum proses ekstraksi informasi dilakukan.

```
    # src/pdf_handler.py:73-93
    # Fungsi ini mendeteksi apakah PDF merupakan hasil scan atau teks
    selectable
    def is_scanned_pdf(pdf_path, char_threshold=50):
        doc = fitz.open(str(pdf_path))
        total_chars = sum(len(page.get_text().strip()) for page in
    doc)
        doc.close()
        result = total_chars < char_threshold    # threshold = 50
    karakter
```

```
    return result
```

Implementasi dilakukan dengan menghitung jumlah karakter yang dapat diekstraksi secara langsung dari dokumen menggunakan PyMuPDF. Jumlah karakter yang diperoleh kemudian dibandingkan dengan nilai ambang batas (threshold) yang telah ditentukan. Apabila jumlah karakter berada di bawah ambang batas, dokumen diasumsikan sebagai scanned PDF dan akan diproses menggunakan PaddleOCR. Sebaliknya, apabila jumlah karakter melebihi nilai ambang batas, dokumen dikategorikan sebagai pure PDF sehingga teks dapat diekstraksi secara langsung tanpa memerlukan proses OCR. Pendekatan ini dipilih karena sederhana, memiliki biaya komputasi yang rendah, serta mampu memberikan hasil klasifikasi dokumen yang memadai pada dataset penelitian.

4.3.3.2 Ekstraksi Pure PDF dengan Crop Header/Footer

Pada dokumen administratif, bagian header dan footer umumnya berisi informasi institusi yang bersifat statis, seperti logo, alamat, nomor telepon, maupun informasi kontak lainnya. Keberadaan informasi tersebut dapat menyebabkan model mempelajari pola yang tidak relevan dengan tujuan identifikasi entitas dinamis. Oleh karena itu, dilakukan proses pemotongan area header dan footer sebelum teks diekstraksi dari dokumen.

```
# src/pdf_handler.py:99-145
# Ekstraksi teks dari pure PDF dengan cropping header/footer
for page_num, page in enumerate(doc, start=1):
    if skip_header_footer and (header_ratio > 0 or footer_ratio > 0):
        rect = page.rect                    # fitz.Rect(x0, y0, x1, y1)
        page_height = rect.height
        y_top = rect.y0 + page_height * header_ratio     # header_ratio = 0.12
        y_bottom = rect.y1 - page_height * footer_ratio     # footer_ratio = 0.10
        clip = fitz.Rect(rect.x0, y_top, rect.x1, y_bottom)

    raw_text = page.get_text("text", sort=True, clip=clip)
```

Implementasi dilakukan dengan menentukan area ekstraksi berdasarkan proporsi tinggi halaman. Bagian atas dokumen dipotong sebesar 12% dari tinggi halaman, sedangkan bagian bawah dipotong sebesar 10%. Nilai tersebut diperoleh berdasarkan observasi terhadap struktur dokumen administratif yang digunakan

pada penelitian ini, di mana sebagian besar kop surat berada pada rentang 10–15% bagian atas halaman dan informasi footer berada pada rentang 8–12% bagian bawah halaman. Melalui proses ini, teks yang diekstraksi menjadi lebih fokus pada isi utama dokumen sehingga dapat meningkatkan kualitas data yang digunakan pada tahap pemrosesan berikutnya.

4.3.3.3 OCR untuk Scanned PDF

Dokumen hasil pemindaian tidak menyimpan informasi teks secara langsung, melainkan berupa citra digital yang merepresentasikan halaman dokumen. Kondisi ini menyebabkan metode ekstraksi teks konvensional tidak dapat digunakan. Oleh karena itu, diperlukan teknologi *Optical Character Recognition* (OCR) untuk mengubah informasi visual menjadi representasi teks yang dapat diproses lebih lanjut oleh sistem.

```
        # src/pdf_handler.py:152-240
        # Proses OCR untuk scanned PDF: render, crop, resize, lalu OCR
        zoom = dpi / 72.0  # dpi = 200, default PDF adalah 72 DPI
        mat  = fitz.Matrix(zoom, zoom)
        pix = page.get_pixmap(matrix=mat, alpha=False)
        img = Image.open(io.BytesIO(img_bytes)).convert("RGB")

        # Crop header/footer dari gambar
        w, h = img.size
        y_top    = int(h * header_ratio)
        y_bottom = int(h * (1 - footer_ratio))
        img = img.crop((0, y_top, w, y_bottom))

        # Resize gambar jika terlalu besar (maks 4096px)
        if max(img.size) > max_img_size:
            ratio = max_img_size / max(img.size)  # max_img_size = 4096
            new_size = (int(img.width * ratio), int(img.height * ratio))
            img = img.resize(new_size, Image.LANCZOS)

        # Jalankan OCR dengan PaddleOCR
        ocr_result = ocr.ocr(img_array, cls=True)
        page_text, confidences = _parse_paddle_result(ocr_result)
avg_conf = sum(confidences) / len(confidences) if confidences else 0.0
```

Implementasi OCR pada penelitian ini menggunakan PaddleOCR yang telah dioptimalkan untuk pengenalan teks berbahasa Indonesia. Sebelum proses OCR dilakukan, setiap halaman PDF dirender menjadi gambar dengan resolusi 200 DPI untuk meningkatkan keterbacaan karakter. Selanjutnya dilakukan proses cropping pada area header dan footer guna menghilangkan informasi yang bersifat berulang

dan tidak relevan terhadap identifikasi entitas. Hasil OCR berupa teks mentah kemudian diteruskan ke tahap prapemrosesan. Penggunaan PaddleOCR dipilih karena memiliki tingkat akurasi yang baik pada dokumen administratif serta mampu menangani variasi kualitas hasil pemindaian.

4.3.3.4 Cleaning Text

Teks hasil ekstraksi dari PDF maupun OCR umumnya masih mengandung berbagai karakter yang tidak diperlukan, seperti spasi berlebih, karakter kontrol, simbol khusus, maupun inkonsistensi format penulisan. Kondisi tersebut dapat memengaruhi kualitas representasi teks yang diterima model sehingga diperlukan proses pembersihan dan normalisasi.

```
    # src/pdf_handler.py:271-277
    # Normalisasi whitespace pada teks hasil ekstraksi
    def _clean_text(text):
        text = text.replace("\r\n", "\n").replace("\r", "\n")
        text = re.sub(r"\n{3,}", "\n\n", text)       # max 2 newline
    berturut-turut
        text = re.sub(r"[ \t]+", " ", text)        # spasi ganda

    return text.strip()
```

Proses normalisasi dilakukan dengan menghapus karakter yang tidak relevan, menyederhanakan penggunaan spasi, serta memperbaiki struktur teks agar lebih konsisten. Selain itu, beberapa pola penulisan yang memiliki variasi format disesuaikan ke dalam bentuk yang seragam. Tahap ini bertujuan untuk mengurangi *noise* pada data sehingga model dapat lebih fokus mempelajari informasi yang benar-benar merepresentasikan isi dokumen. Dengan kualitas data yang lebih baik, performa model dalam mengenali entitas administratif dapat ditingkatkan.

4.3.3.5 BIO Tagging Scheme

Model Named Entity Recognition memerlukan data pelatihan yang telah diberikan anotasi dalam format tertentu. Pada penelitian ini digunakan skema BIO (Beginning, Inside, Outside) yang merupakan salah satu standar anotasi paling umum dalam tugas pengenalan entitas.

```
    # config.py:56-64
```

```
        # Skema BIO tagging: B-<label>, I-<label>, O
        # Dengan 9 label entitas, total kelas BIO = 1 O + 9 B + 9 I = 19
        kelas
        BIO_LABELS = ["O"] + [
            prefix + label for label in LABELS
            for prefix in ("B-", "I-")
        ]
        LABEL2ID = {lbl: i for i, lbl in enumerate(BIO_LABELS)}
        ID2LABEL = {i: lbl for lbl, i in LABEL2ID.items()}

NUM_LABELS = len(BIO_LABELS)  # 19 kelas
```

Pada skema BIO, token pertama dari suatu entitas diberi label B (*Beginning*), token berikutnya dalam entitas yang sama diberi label I (*Inside*), sedangkan token yang tidak termasuk entitas apa pun diberi label O (*Outside*). Pendekatan ini memungkinkan model memahami batas awal dan akhir suatu entitas secara lebih akurat. Penggunaan format BIO juga kompatibel dengan arsitektur transformer seperti IndoBERT yang digunakan pada penelitian ini sehingga mempermudah proses pelatihan dan evaluasi model.

4.3.3.6 Pembuatan BIO Tags dari Anotasi

Model transformer tidak memproses teks dalam bentuk kata utuh, melainkan dalam bentuk token yang dihasilkan melalui proses tokenisasi. Akibatnya, satu kata dapat dipecah menjadi beberapa sub-token sehingga diperlukan mekanisme penyelarasan antara token hasil tokenisasi dan label BIO yang telah dibuat sebelumnya.

```
        # src/dataset.py:449-535
        #  Konversi entity annotations ke BIO tags menggunakan spaCy
        tokenizer
        def create_bio_tags(text, entities):
            nlp = _get_nlp()
            doc = nlp(text)
            tokens = [tok.text for tok in doc]
            char_tags = ["O"] * len(text)

            # Urutkan entity terpanjang dulu agar label spesifik overwrite
        label umum
            sorted_ents = sorted(
                [e for e in entities if e["label"] in LABELS and
        e["label"] not in NER_EXCLUDE_LABELS],
                key=lambda e: (-(e["end"] - e["start"]),),
            )
            for ent in sorted_ents:
                start = ent["start"]
                end   = ent["end"]
                label = ent["label"]
                for i in range(start, end):
                    if char_tags[i] != "O":
                        continue  # Jangan overwrite label yang sudah ada
```

```
                    if i == start:
                        char_tags[i] = f"B-{label}"
                    else:
                        char_tags[i] = f"I-{label}"

        # Map char tags ke token tags (ambil tag dari karakter pertama
    token)
        bio_tags = []
        for tok in doc:
            token_real_start = text.find(tok.text)
            if token_real_start == -1:
                bio_tags.append("O")
            else:
                bio_tags.append(char_tags[token_real_start])
    return tokens, bio_tags
```

Proses alignment dilakukan dengan mempertahankan label asli pada token pertama hasil pemecahan kata, sementara token-token berikutnya memperoleh label yang sesuai dengan aturan BIO. Pendekatan ini memastikan bahwa informasi anotasi tidak hilang selama proses tokenisasi. Dengan demikian, model tetap dapat mempelajari hubungan antara token dan label entitas secara konsisten meskipun terjadi pemecahan kata menjadi beberapa sub-token.

4.3.3.7 Alignment Sub-word Token untuk BERT

Model BERT/IndoBERT menggunakan sub-word tokenizer (WordPiece) yang dapat memecah sebuah kata menjadi beberapa token (misal: “bertugas” → “bert”, “##ugas”). Agar label BIO tetap sesuai, diperlukan penyelarasan (alignment) sehingga hanya token pertama dari suatu kata yang mempertahankan label asli, sedangkan sub-token berikutnya diberi label khusus (biasanya I- atau diabaikan dalam loss). Proses ini memastikan model belajar pada tingkat sub-token yang benar tanpa menghitung prediksi pada pecahan kata sebagai kesalahan terpisah.

```
    # src/dataset.py:541-581
    # Tokenisasi IndoBERT dan alignment BIO labels ke sub-word tokens
    def tokenize_and_align_labels(tokens, bio_tags, max_length=512):
        tokenizer = get_tokenizer()
        tokenized = tokenizer(
            tokens, is_split_into_words=True,
            truncation=True, padding="max_length",
            max_length=max_length,
        )
        word_ids = tokenized.word_ids()
        label_ids = []
        prev_word = None
        for wid in word_ids:
            if wid is None:
```

```
                label_ids.append(-100)      #   [CLS],   [SEP],   [PAD]
    diabaikan
            elif wid != prev_word:
                label_ids.append(LABEL2ID.get(bio_tags[wid],   0))     #
    token pertama kata
            else:
                # Sub-word berikutnya: set I-label
                current_tag = bio_tags[wid]
                if current_tag.startswith("B-"):
                    current_tag = "I-" + current_tag[2:]
                label_ids.append(LABEL2ID.get(current_tag, -100))
            prev_word = wid
        tokenized["labels"] = label_ids

    return dict(tokenized)
```

Fungsi yang digunakan melakukan tokenisasi dengan tokenizer IndoBERT dalam mode is_split_into_words=True. Setelah tokenisasi, fungsi mengambil pemetaan word_ids yang menghubungkan setiap sub-token ke indeks kata asli. Untuk setiap sub-token, jika merupakan token khusus (CLS, SEP, PAD), label diisi -100 agar diabaikan. Jika sub-token adalah yang pertama dari suatu kata, label asli dari BIO tags diambil. Jika sub-token lanjutan, label diubah menjadi I-... (bila aslinya B-...) agar tetap menjadi bagian dari entitas yang sama. Hasil akhir berupa dictionary berisi input_ids, attention_mask, dan labels yang siap dimasukkan ke model.

4.3.3.8 Split Dataset

Pembagian dataset merupakan tahapan penting dalam pengembangan model pembelajaran mesin karena digunakan untuk mengukur kemampuan model dalam melakukan generalisasi terhadap data yang belum pernah dilihat sebelumnya.

```
    # src/dataset.py:625-646
    # Split dataset secara acak menjadi train/val/test
    def    split_dataset(samples,    train_ratio=0.8,    val_ratio=0.1,
    test_ratio=0.1, seed=42):
        random.seed(seed)
        shuffled = samples.copy()
        random.shuffle(shuffled)
        n = len(shuffled)
        n_train = int(n * train_ratio)
        n_val = int(n * val_ratio)
        train = shuffled[:n_train]
        val = shuffled[n_train : n_train + n_val]
        test = shuffled[n_train + n_val :]

    return train, val, test
```

Dataset yang telah melalui proses anotasi dan prapemrosesan dibagi menjadi tiga kelompok, yaitu data pelatihan (*training set*), data validasi (*validation set*), dan

data pengujian (*test set*). Data pelatihan digunakan untuk memperbarui parameter model, data validasi digunakan untuk memantau performa selama proses pelatihan, sedangkan data pengujian digunakan untuk mengukur performa akhir model secara objektif. Pembagian ini dilakukan untuk mengurangi risiko *overfitting* serta memastikan bahwa hasil evaluasi merepresentasikan kemampuan model pada kondisi nyata.

4.3.3.2 Experimen Fine Tuning

dilakukan beberapa eksperimen fine-tuning menggunakan berbagai strategi pelatihan. Strategi yang diuji meliputi Full Fine-Tuning, Layer-wise Learning Rate Decay (LLRD), serta Progressive Unfreezing. Setiap eksperimen menghasilkan nilai evaluasi yang berbeda sehingga dapat dibandingkan untuk menentukan konfigurasi terbaik.

4.3.4 Perancangan Arsitektur Model

Arsitektur sistem terdiri dari beberapa komponen yang bekerja secara pipeline: spaCy untuk tokenisasi kalimat, IndoBERT untuk Named Entity Recognition, LayoutLMv3 untuk deteksi tabel, dan rule-based postprocessing untuk ekstraksi PERIHAL dan ISI dokumen.

4.3.4.1 Pipeline NER Utama

Pipeline utama NER menerima teks dokumen mentah dan mengembalikan kamus entitas yang ditemukan. Proses dimulai dengan tokenisasi kalimat menggunakan spaCy, kemudian teks panjang dibagi menjadi potongan (chunk) maksimal 400 token dengan teknik sliding window agar sesuai batas 512 token model IndoBERT. Setiap chunk diprediksi secara independen, lalu hasilnya digabungkan. Setelah prediksi NER, dilakukan post-processing dengan rule-based extraction untuk entitas PERIHAL (menggunakan regex) dan ISI (menggunakan body finder), serta deteksi tabel menggunakan LayoutLMv3 jika file PDF tersedia.

```
# src/inference.py:565-644
```

```
    # Main NER pipeline: teks dokumen -> entity dict
    def run_ner(text, chunk_size=400, pdf_path=None):
        _ensure_model()
        nlp = _get_nlp()
        doc = nlp(text)

        # Sliding window per kalimat dengan chunk_size token
        all_spans = []
        chunk_tokens = []
        for sent in doc.sents:
            sent_tokens = [tok.text for tok in sent if not
tok.is_space]
            if len(chunk_tokens) + len(sent_tokens) > chunk_size:
                _flush_chunk()
            chunk_tokens.extend(sent_tokens)
        _flush_chunk()

        entities = aggregate_entities(all_spans)

        # Post-processing PERIHAL dengan regex
        perihal = _extract_perihal_from_text(text)
        if perihal:
            entities["PERIHAL"] = perihal

        # Ekstrak ISI dengan rule-based body finder
        isi = _extract_isi_from_text(text, known_entities=entities)
        if isi:
            entities["ISI"] = isi

        # Deteksi TABEL dengan LayoutLMv3
        tabel_content = _extract_table_content(text,
pdf_path=pdf_path)
        if tabel_content:
            entities["TABEL"] = tabel_content

    return entities
```

Fungsi utama memuat model IndoBERT dan spaCy, lalu memecah teks menjadi kalimat. Kalimat dikumpulkan ke dalam chunk hingga melebihi ukuran maksimum, kemudian setiap chunk diprediksi oleh model. Hasil prediksi dari semua chunk diagregasi. Setelah itu, fungsi ekstraksi PERIHAL dijalankan dengan mencocokkan pola regex seperti Perihal: atau Hal:, dan fungsi ekstraksi ISI mencari kalimat pembuka isi surat yang khas dalam bahasa Indonesia. Jika file PDF disediakan, deteksi tabel menggunakan LayoutLMv3 juga dijalankan. Hasil akhir berupa kamus yang berisi semua entitas yang berhasil diekstrak

4.3.4.2 Arsitektur Model IndoBERT

Model yang digunakan untuk NER adalah BertForTokenClassification dari HuggingFace dengan encoder IndoBERT (indobenchmark/indobert-base-p1).

Arsitektur ini terdiri dari 12 lapisan transformer yang menghasilkan representasi kontekstual untuk setiap token, kemudian sebuah classification head (lapisan linear) memetakan representasi tersebut ke 19 kelas BIO (9 entitas dikali 2 (B dan I) ditambah 1 kelas O). Model diinisialisasi dengan konfigurasi yang menentukan jumlah label, dropout (biasanya 0,1), serta pemetaan antara ID label dan nama label.

```
# src/model.py:28-73
# Membangun model IndoBERT untuk token classification (NER)
def build_model(pretrained, num_labels, dropout, from_checkpoint=None):
    if from_checkpoint:
        model = AutoModelForTokenClassification.from_pretrained(from_checkpoint)
    else:
        config = AutoConfig.from_pretrained(
            pretrained, num_labels=num_labels,
            id2label=ID2LABEL, label2id=LABEL2ID,
            hidden_dropout_prob=dropout,
            attention_probs_dropout_prob=dropout,
        )
        model = AutoModelForTokenClassification.from_pretrained(pretrained, config=config)
    total_params = sum(p.numel() for p in model.parameters())
    trainable_params = sum(p.numel() for p in model.parameters() if p.requires_grad)
return model
```

Fungsi build_model menerima nama pretrained model, jumlah label, nilai dropout, dan opsi checkpoint. Jika checkpoint disediakan, model langsung dimuat dari checkpoint tersebut. Jika tidak, fungsi memuat konfigurasi dari model pretrained dan memperbarui num_labels, id2label, label2id, serta nilai dropout. Kemudian AutoModelForTokenClassification.from_pretrained dipanggil untuk membuat model dengan konfigurasi yang telah disesuaikan. Setelah model dibuat, fungsi menghitung jumlah total parameter dan parameter yang dapat dilatih, lalu mengembalikan model tersebut.

4.3.4.3 Sliding Window Prediction

Karena model IndoBERT memiliki batas maksimum 512 token, teks dokumen yang panjang tidak dapat diproses secara langsung. Metode sliding window digunakan untuk memotong teks menjadi potongan-potongan yang saling tumpang tindih (overlap) agar tidak ada informasi yang hilang di batas potongan. Setiap potongan diprediksi secara independen, kemudian prediksi pada token yang

sama dari potongan yang berbeda (pada area tumpang tindih) dapat digabungkan, misalnya dengan mengambil suara mayoritas atau prediksi dengan kepercayaan tertinggi.

```
    # src/inference.py:141-189
    # Prediksi NER dengan sliding window untuk teks panjang
    def _predict_tokens(tokens, stride=64, max_len=512):
        encoding = tokenizer(
            Tokens, is_split_into_words=True,
            truncation=True, padding="max_length",
            max_length=max_len, return_tensors="pt",
        )
        word_ids = encoding.word_ids()
        with torch.no_grad():
            outputs = model(
                input_ids=encoding["input_ids"].to(device),
                attention_mask=encoding["attention_mask"].to(device),
            )
        logits   = outputs.logits[0].cpu()
        pred_ids = logits.argmax(dim=-1).numpy()  # token -> BIO tag

        # Ambil satu prediksi per kata (sub-word pertama)
        word_preds = {}
        prev_wid = None
        for idx, wid in enumerate(word_ids):
            if wid is None or wid == prev_wid:
                prev_wid = wid
                continue
            word_preds[wid] = ID2LABEL.get(int(pred_ids[idx]), "O")
            prev_wid = wid

    return [word_preds.get(i, "O") for i in range(len(tokens))]
```

Fungsi prediksi menerima daftar token dan melakukan tokenisasi dengan padding serta truncation ke panjang maksimum 512. Model melakukan inferensi tanpa menghitung gradien. Logits yang dihasilkan diambil nilai maksimumnya untuk mendapatkan ID kelas BIO. Selanjutnya, fungsi menyelaraskan kembali prediksi dengan kata asli: hanya token pertama dari setiap kata yang diambil prediksinya, karena sub-token berikutnya hanya mengulang informasi. Hasilnya adalah daftar label BIO untuk setiap kata dalam urutan aslinya.

4.3.4.4 Ekstraksi Entity Spans dari BIO Tags

Hasil prediksi model berupa urutan label BIO per kata. Untuk mendapatkan entitas yang utuh (misalnya "001/SK/2024" sebagai satu entitas NOMOR_SURAT), urutan tersebut harus dikonversi menjadi rentang (spans) entitas. Aturannya: setiap label B-label menandai awal entitas baru, label I-label yang mengikuti (dengan label yang sama) menjadi bagian dari entitas yang sedang

berjalan, sedangkan label O atau perubahan ke label lain menutup entitas sebelumnya. Proses ini menghasilkan daftar entitas yang masing-masing berisi label dan nilai teks gabungan.

```
    # src/inference.py:195-230
    # Konversi BIO token-tag sequence ke list entitas
    def extract_spans(tokens, bio_tags):
        spans, current_label, current_tokens = [], None, []
        for token, tag in zip(tokens, bio_tags):
            if tag.startswith("B-"):
                if current_label and current_tokens:
                    spans.append({"label":  current_label,  "value":  "
".join(current_tokens)})
                current_label  = tag[2:]
                current_tokens = [token]
            elif tag.startswith("I-") and current_label:
                label = tag[2:]
                if label == current_label:
                    current_tokens.append(token)
                else:
                    spans.append({"label":  current_label,  "value":  "
".join(current_tokens)})
                    current_label  = label
                    current_tokens = [token]
            else:
                if current_label and current_tokens:
                    spans.append({"label":  current_label,  "value":  "
".join(current_tokens)})
                current_label, current_tokens = None, []
        if current_label and current_tokens:
            spans.append({"label":    current_label,    "value":    "
".join(current_tokens)})

    return spans
```

Fungsi extract_spans mengiterasi pasangan token dan label BIO. Ketika menemukan label yang diawali dengan B-, fungsi menyelesaikan entitas sebelumnya (jika ada) dan memulai entitas baru dengan label yang diambil dari karakter ke-2 hingga akhir. Token pertama disimpan. Jika label adalah I- dan label tersebut cocok dengan label entitas yang sedang berjalan, token ditambahkan ke entitas tersebut. Jika label I- tetapi labelnya berbeda, entitas sebelumnya ditutup dan entitas baru dimulai. Label O atau akhir iterasi akan menutup entitas yang sedang berjalan. Hasilnya adalah daftar spans yang berisi label dan teks entitas.

4.3.4.5 Rule-based PERIHAL

Entitas PERIHAL (perihal surat) memiliki pola yang relatif tetap dalam dokumen administratif bahasa Indonesia, biasanya diawali dengan kata "Perihal:" atau "Hal:" diikuti dengan teks singkat. Karena pola ini mudah diekstrak dengan

ekspresi reguler (regex), pendekatan rule-based lebih akurat dan cepat dibandingkan mengandalkan model NER murni. Fungsi ini mencari pola tersebut dalam teks dokumen dan mengembalikan teks perihal yang ditemukan.

```
    # src/inference.py:263-292
    # Ekstrak PERIHAL dari dokumen dengan regex pattern
    def _extract_perihal_from_text(text):
        patterns = [
            r'(?:perihal|hal|topik|subject)\s*:\s*([^\n]+)',
            r'(?:perihal|hal|topik|subject)\s*\n\s*:?\s*([^\n]+)',
        ]
        for pat in patterns:
            match = re.search(pat, text, re.IGNORECASE)
            if match:
                perihal_text = match.group(1).strip()
                if perihal_text and len(perihal_text) > 2:
                    return _validate_perihal(perihal_text)
    return None
```

Fungsi ekstrak PERIHAL mendefinisikan beberapa pola regex, antara lain mencari kata "perihal", "hal", "topik", atau "subject" yang diikuti spasi opsional, tanda titik dua, spasi, lalu menangkap semua karakter hingga akhir baris. Pola-pola tersebut diiterasi, dan jika ditemukan kecocokan (dengan pengabaian huruf besar-kecil), teks yang tertangkap dibersihkan. Jika panjang teks lebih dari dua karakter, fungsi validasi dipanggil untuk memastikan bahwa yang tertangkap bukanlah bagian dari entitas PENERIMA (misalnya "Yth."). Hasilnya dikembalikan, atau None jika tidak ditemukan.

4.3.4.6 Rule-based ISI (Body Text)

Isi utama (body) dokumen administratif dimulai setelah bagian pembuka (kop surat, nomor surat, perihal) dan sebelum bagian penutup (tanda tangan, lampiran, tembusan). Karena model NER dilatih untuk entitas pendek, bukan paragraf panjang, pendekatan rule-based digunakan untuk mengekstraksi body. Fungsi ini mencari kalimat pembuka yang khas dalam surat resmi bahasa Indonesia (misal: "Sehubungan dengan...", "Bersama ini...", "Dengan ini kami menginformasikan...") dan mengambil beberapa paragraf berikutnya hingga ditemukan indikator penutup.

```
    # src/inference.py:344-522
    # Ekstrak isi utama dokumen dengan rule-based body finder
    _BODY_OPENER = re.compile(
        r'^(?:'

    r'kami\s+(?:menginformasikan|memberitahukan|sampaikan|ajak|undang|selen
```

```
        ggarakan)|'
            r'bersama\s+ini|sehubungan\s+dengan|menindaklanjuti|'
            r'berkenaan\s+dengan|dalam\s+rangka|berdasarkan|'
            r'yang\s+bertanda\s+tangan|menerangkan\s+bahwa|'
            r'sesuai\s+dengan|diberitahukan\s+bahwa|'
            r'dengan\s+ini\s+(?:kami|memberitahukan|mengumumkan)|'
            r'melalui\s+surat\s+ini'
            r')', re.IGNORECASE,
    )
```

Fungsi ekstrak ISI menggunakan kompilasi regex dari berbagai pola pembuka isi surat (misalnya: "kami menginformasikan", "sehubungan dengan", "yang bertanda tangan", "menerangkan bahwa", dan lain-lain). Fungsi mencari kecocokan pola tersebut di awal teks. Setelah ditemukan, fungsi mengumpulkan paragraf-paragraf berikutnya hingga mencapai batas maksimum karakter (800) atau menemukan kalimat penutup seperti "Demikian surat ini dibuat...". Paragraf yang terkumpul kemudian digabungkan dan dikembalikan sebagai isi dokumen.

4.3.4.7 Deteksi Tabel dengan LayoutLMv3

Dokumen administratif sering kali menyertakan tabel (misalnya daftar peserta, rincian biaya, jadwal kegiatan). Untuk mengekstrak informasi dari tabel, diperlukan pemahaman tata letak spasial selain teks. LayoutLMv3 adalah model transformer yang dirancang khusus untuk memahami dokumen dengan menggabungkan teks, posisi bounding box, dan gambar. Fungsi ini memanggil modul table_detector yang mengimplementasikan deteksi tabel hybrid (menggabungkan aturan spasial dengan LayoutLMv3). Jika file PDF tersedia, bounding boxes aktual digunakan untuk meningkatkan akurasi.

```
        # src/inference.py:530-559
        # Deteksi dan ekstrak konten tabel menggunakan LayoutLMv3
        def _extract_table_content(text, pdf_path=None):
            from src.table_detector import hybrid_detect, spans_to_text,
        DEFAULT_LAYOUTLM_MODEL
            mp = model_path or DEFAULT_LAYOUTLM_MODEL
            spans = hybrid_detect(
                text=text, pdf_path=pdf_path,
                model_path=mp, min_confidence=0.50,
            )
            if spans:
                combined = spans_to_text(spans)
                return combined if combined.strip() else None

    return None
```

Fungsi ekstrak tabel mengimpor fungsi hybrid_detect dan spans_to_text dari modul table detector, serta konstanta jalur model LayoutLMv3 yang telah dilatih. Fungsi memanggil hybrid_detect dengan parameter teks, jalur PDF opsional, jalur model, dan ambang kepercayaan minimum 0,50. Hasil deteksi berupa daftar span (region tabel beserta teksnya). Jika ada hasil, fungsi spans_to_text menggabungkan teks dari semua span menjadi satu string. String tersebut dikembalikan jika tidak kosong, selain itu mengembalikan None.

4.3.5 Konfigurasi dan Finetune Model

Tahap ini mencakup konfigurasi hyperparameter model dan strategi fine-tuning yang digunakan untuk melatih IndoBERT pada dataset NER surat Indonesia.

4.3.5.1 Model Configuration

Konfigurasi model menentukan pretrained model yang digunakan, panjang maksimum input token, jumlah label, dan nilai dropout. Kelas konfigurasi menggunakan dekorator dataclass dari Python untuk menyederhanakan pendefinisian atribut. Parameter pretrained_model diisi dengan model IndoBERT dasar yang telah dilatih pada korpus bahasa Indonesia. max_length diatur ke 512 sesuai batasan arsitektur BERT. num_labels diambil dari konstanta total kelas BIO (19). Nilai dropout 0,1 digunakan untuk regularisasi guna mencegah overfitting.

```
    # config.py:70-77
    # Konfigurasi model IndoBERT untuk NER
    @dataclass
    class ModelConfig:
        pretrained_model: str  = "indobenchmark/indobert-base-p1"
        max_length: int        = 512
        num_labels: int        = NUM_LABELS  # 19 kelas BIO
        dropout: float         = 0.1

fine_tuned_path: str   = str(CKPT_DIR / "indobert-ner-finetuned")
```

Kelas ModelConfig mendefinisikan atribut-atribut seperti pretrained_model, max_length = 512, num_labels = NUM_LABELS, dropout = 0.1, dan fine_tuned_path yang merupakan jalur direktori tempat model hasil fine-tuning akan disimpan. Konfigurasi ini memudahkan perubahan hyperparameter tanpa mengubah kode utama.

4.3.5.2 Training Configuration

Konfigurasi pelatihan mencakup hyperparameter seperti ukuran batch, jumlah epoch, laju pembelajaran (learning rate), weight decay, warmup ratio, dan penggunaan FP16 (presisi 16-bit) untuk menghemat memori GPU. Kelas konfigurasi juga menentukan proporsi pembagian data (train 80 persen, val 10 persen, test 10 persen) dan nilai seed untuk reproduksibilitas.

```
# config.py:82-98
# Konfigurasi hyperparameter training
@dataclass
class TrainConfig:
    batch_size: int          = 16      # batch 16 untuk training
    eval_batch_size: int     = 16
    epochs: int              = 10
    learning_rate: float     = 2e-5
    weight_decay: float      = 0.01
    warmup_ratio: float      = 0.1
    gradient_clip: float     = 1.0
    seed: int                = 42
    train_split: float       = 0.8
    val_split: float         = 0.1
    # test_split = 0.1 (sisa)

fp16: bool              = True    # Hemat ~50% VRAM di GPU NVIDIA
```

Kelas TrainConfig mendefinisikan batch_size = 16 untuk pelatihan dan evaluasi. epochs = 10 sebagai jumlah iterasi penuh atas data pelatihan. learning_rate = 2e-5 adalah nilai umum untuk fine-tuning BERT. weight_decay = 0.01 untuk regularisasi L2. warmup_ratio = 0.1 berarti 10 persen dari total langkah digunakan untuk menaikkan laju pembelajaran secara linear. gradient_clip = 1.0 mencegah exploding gradient. seed = 42 memastikan hasil acak dapat direproduksi. fp16 = True mengaktifkan pelatihan dengan presisi 16-bit yang dapat menghemat sekitar 50 persen memori VRAM di GPU NVIDIA.

4.3.5.3 Label Smoothing Cross-Entropy Loss

Loss function standar untuk klasifikasi token adalah cross-entropy. Namun, cross-entropy dapat menyebabkan model menjadi terlalu percaya diri (overconfident), yang berpotensi menurunkan kemampuan generalisasi. Label smoothing adalah teknik yang menggantikan target one-hot dengan distribusi yang lebih halus, memberikan sebagian bobot (epsilon) ke kelas-kelas lain secara seragam. Dengan

demikian, model tidak terdorong untuk memprediksi probabilitas 1 pada kelas target, sehingga lebih robust terhadap noise dan mengurangi overfitting.

```
 #src/finetune_indobert.py:60-86
# Cross-entropy dengan label smoothing (epsilon = 0.1)
class LabelSmoothingCrossEntropy(nn.Module):
    def __init__(self, smoothing=0.1, ignore_index=-100):
        super().__init__()
        self.smoothing    = smoothing
        self.ignore_index = ignore_index
    def forward(self, logits, targets):
        n_cls = logits.size(-1)
        flat_logits  = logits.view(-1, n_cls)
        flat_targets = targets.view(-1)
        mask = flat_targets != self.ignore_index
        flat_logits  = flat_logits[mask]
        flat_targets = flat_targets[mask]
        log_probs = F.log_softmax(flat_logits, dim=-1)
        nll                =  F.nll_loss(log_probs,  flat_targets,
reduction="mean")
        smooth    = -log_probs.mean()
        # Loss = (1 - epsilon) * NLL + epsilon * smooth_loss
 return (1.0 - self.smoothing) * nll + self.smoothing * smooth
```

Kelas LabelSmoothingCrossEntropy mewarisi nn.Module dan diinisialisasi dengan parameter smoothing (default 0,1) dan ignore_index (default -100). Pada metode forward, logits dan target diterima. Logits di-reshape menjadi dua dimensi, target di-flatten. Mask dibuat untuk mengabaikan token dengan label ignore_index. Log probabilitas dihitung dengan log_softmax. Negative log likelihood (NLL) dihitung untuk token yang tidak diabaikan. Smooth loss adalah negatif dari rata-rata log probabilitas. Hasil akhir adalah (1 - smoothing) dikali NLL ditambah smoothing dikali smooth_loss. Dengan smoothing 0,1, 10 persen bobot diberikan pada distribusi seragam.

4.3.5.4 Optimizer dan Learning Rate Scheduler

Optimizer yang digunakan adalah AdamW (Adam dengan weight decay yang benar). Learning rate scheduler yang umum pada fine-tuning BERT adalah linear schedule with warmup: laju pembelajaran meningkat secara linear dari 0 hingga target LR selama sejumlah langkah warmup, kemudian menurun secara linear kembali ke 0 hingga akhir pelatihan. Strategi ini membantu stabilitas pelatihan di awal (warmup) dan memungkinkan konvergensi yang baik di akhir.

```
# src/trainer.py:168-187
# Setup optimizer AdamW dan linear warmup scheduler
```

```
        optimizer       =       AdamW(param_groups,       lr=learning_rate,
        weight_decay=weight_decay)
        total_steps  = len(train_loader) * epochs
        warmup_steps  =  int(total_steps  *  warmup_ratio)   #  warmup_ratio  =
        0.1
        scheduler = get_linear_schedule_with_warmup(
            optimizer,                          num_warmup_steps=warmup_steps,
        num_training_steps=total_steps

)
```

Optimizer AdamW dibuat dengan parameter kelompok yang biasanya memisahkan weight decay untuk bias dan LayerNorm (diberi weight decay 0). Laju pembelajaran awal dan weight decay disediakan. Total langkah pelatihan dihitung sebagai panjang train loader dikali jumlah epoch. Jumlah langkah warmup adalah total langkah dikali warmup ratio. Scheduler linear warmup kemudian dibuat dengan optimizer, num_warmup_steps, dan num_training_steps. Selama pelatihan, scheduler dipanggil setiap langkah untuk memperbarui laju pembelajaran.

4.3.6 Pelaksanaan Eksperimen Pelatihan

Telah dilakukan serangkaian eksperimen pelatihan sebanyak 13 kali menggunakan berbagai strategi fine-tuning. Eksperimen dibagi ke dalamempat fase utama, yaitu baseline dan strategi dasar, eksplorasi Full Fine-Tuning dan Layer-wise Learning Rate Decay (LLRD), optimasi Progressive Unfreezing, sertaeksplorasi lanjutan menggunakan pendekatan hybrid. Setiap eksperimendievaluasi menggunakan metrik precision, recall, dan F1-score pada data pengujian.

1. Run ID #1

Run pertama digunakan sebagai baseline dengan menerapkan strategi Full Fine-Tuning selama 20 epoch. Pada konfigurasi ini seluruh parameter model IndoBERT diperbarui selama proses pelatihan. Hasil evaluasi menunjukkan nilai F1-score sebesar 0.7593, precision 0.6649, dan recall 0.8849. Nilai recall yang relatif tinggi menunjukkan bahwa model mampu mengenali sebagian besar entitas yang ada pada data pengujian, meskipun masih terdapat sejumlah prediksi yang kurang tepat sehingga precision berada pada nilai yang lebih rendah. Hasil ini

menjadi acuan untuk membandingkan efektivitas konfigurasi pada eksperimen selanjutnya.

2. Run ID #2

Eksperimen kedua menggunakan konfigurasi Full Fine-Tuning selama 10 epoch. Hasil evaluasi menunjukkan penurunan performa yang cukup signifikan dengan F1-score sebesar 0.5455, precision 0.4180, dan recall 0.7846. Penurunan ini mengindikasikan bahwa jumlah epoch yang lebih sedikit belum cukup untuk menghasilkan proses pembelajaran yang optimal. Model masih mampu menemukan sebagian besar entitas, namun tingkat kesalahan prediksi meningkat sehingga precision dan F1-score mengalami penurunan dibandingkan baseline.

3. Run ID #3

Run ketiga juga menggunakan strategi Full Fine-Tuning selama 10 epoch dengan konfigurasi yang berbeda dari run sebelumnya. Model menghasilkan F1-score sebesar 0.6149, precision 0.4908, dan recall 0.8231. Hasil ini menunjukkan peningkatan dibandingkan Run ID #2, namun masih berada di bawah baseline. Nilai recall yang relatif tinggi menunjukkan bahwa model cukup baik dalam mendeteksi entitas, tetapi kemampuan membedakan entitas secara akurat masih perlu ditingkatkan.

4. Run ID #4

Eksperimen keempat merupakan pengujian pertama yang menggunakan strategi Layer-wise Learning Rate Decay (LLRD). Hasil evaluasi menunjukkan F1-score sebesar 0.5450, precision 0.4219, dan recall 0.7692. Performa yang diperoleh hampir sama dengan Run ID #2 dan masih berada di bawah baseline. Hasil ini menunjukkan bahwa konfigurasi LLRD yang digunakan pada eksperimen ini belum mampu memberikan keuntungan yang signifikan terhadap proses adaptasi model pada domain dokumen administratif.

5. Run ID #5

Run kelima menggunakan strategi LLRD dengan konfigurasi yang lebih agresif. Hasil evaluasi menunjukkan penurunan performa yang sangat drastis dengan F1-score sebesar 0.0700, precision 0.0456, dan recall 0.1511. Nilai yang sangat

rendah pada seluruh metrik menunjukkan bahwa model gagal mempelajari pola entitas secara efektif. Kondisi ini mengindikasikan bahwa konfigurasi learning rate yang digunakan terlalu ekstrem sehingga menghambat proses pembelajaran model.

6. Run ID #6

Eksperimen keenam kembali menggunakan strategi Full Fine-Tuning selama 10 epoch. Hasil evaluasi menunjukkan F1-score sebesar 0.4399, precision 0.3212, dan recall 0.6978. Nilai tersebut lebih rendah dibandingkan beberapa eksperimen sebelumnya, yang menunjukkan bahwa konfigurasi yang digunakan belum mampu menghasilkan representasi yang baik untuk tugas Named Entity Recognition pada dokumen administratif.

7. Run ID #7

Run ketujuh menerapkan Full Fine-Tuning selama 14 epoch. Hasil evaluasi menunjukkan peningkatan dibandingkan Run ID #6 dengan F1-score sebesar 0.5047, precision 0.3600, dan recall 0.8438. Meskipun recall yang diperoleh cukup tinggi, precision masih rendah sehingga banyak prediksi yang tidak sesuai dengan label sebenarnya. Akibatnya, performa keseluruhan model masih belum mampu melampaui baseline.

8. Run ID #8

Eksperimen kedelapan menghasilkan peningkatan performa yang signifikan dibandingkan eksperimen sebelumnya. Model memperoleh F1-score sebesar 0.8725, precision 0.8396, dan recall 0.9082 setelah dilatih selama 10 epoch. Hasil ini menunjukkan bahwa konfigurasi yang digunakan mampu meningkatkan keseimbangan antara precision dan recall, sehingga kualitas prediksi entitas menjadi jauh lebih baik dibandingkan baseline.

9. Run ID #9

Run kesembilan dilaksanakan selama 5 epoch dan menghasilkan F1-score sebesar 0.8867, precision 0.8571, dan recall 0.9184. Meskipun jumlah epoch lebih sedikit dibandingkan Run ID #8, performa model justru meningkat. Hal ini menunjukkan

bahwa konfigurasi yang digunakan mampu mempercepat proses konvergensi dan menghasilkan generalisasi yang lebih baik terhadap data pengujian.

10. Run ID #10

Eksperimen kesepuluh menggunakan strategi LLRD dan menghasilkan F1-score sebesar 0.9010, precision 0.8750, serta recall 0.9286. Hasil ini menunjukkan bahwa penerapan LLRD pada konfigurasi yang tepat dapat meningkatkan performa model secara signifikan. Dibandingkan eksperimen LLRD sebelumnya, konfigurasi pada run ini mampu menjaga keseimbangan antara proses adaptasi model dan stabilitas parameter pada setiap layer Transformer.

11. Run ID #11

Run kesebelas kembali menggunakan Full Fine-Tuning selama 6 epoch. Model menghasilkan F1-score sebesar 0.8966, precision 0.8667, dan recall 0.9286. Hasil ini sedikit lebih rendah dibandingkan Run ID #10, namun tetap menunjukkan performa yang sangat baik. Nilai precision dan recall yang tinggi mengindikasikan bahwa model telah mampu mengenali entitas administratif dengan tingkat akurasi yang tinggi.

12. Run ID #12

Eksperimen kedua belas menghasilkan performa terbaik dari seluruh rangkaian pengujian. Model memperoleh F1-score sebesar 0.9100, precision 0.8922, dan recall 0.9286 hanya dalam 4 epoch pelatihan. Hasil ini menunjukkan bahwa konfigurasi yang digunakan mampu mencapai konvergensi secara cepat serta menghasilkan keseimbangan yang baik antara precision dan recall. Dibandingkan dengan baseline pada Run ID #1 yang memperoleh F1-score sebesar 0.7593, konfigurasi ini berhasil meningkatkan performa sebesar 0.1507. Berdasarkan nilai F1-score yang merupakan metrik utama pada tugas Named Entity Recognition, Run ID #12 menjadi konfigurasi dengan performa terbaik dalam penelitian ini.

13. Run ID #13

Run ketiga belas dilakukan sebagai eksperimen lanjutan untuk mengevaluasi konsistensi performa model pada jumlah epoch yang lebih besar. Model dilatih selama 13 epoch dan menghasilkan F1-score sebesar 0.8856, precision 0.8641,

serta recall 0.9082. Meskipun nilai F1-score yang diperoleh sedikit lebih rendah dibandingkan Run ID #12, performa model tetap berada pada kategori yang tinggi. Hasil ini menunjukkan bahwa model mampu mempertahankan kemampuan prediksi yang baik meskipun menjalani proses pelatihan yang lebih panjang. Pada dataset yang relatif kecil, yaitu sebanyak 246 data, kondisi tersebut mengindikasikan bahwa konfigurasi Run ID #13 memiliki tingkat stabilitas pelatihan yang baik karena tidak mengalami penurunan performa yang drastis setelah penambahan jumlah epoch. Oleh karena itu, Run ID #13 dapat dianggap sebagai konfigurasi yang lebih stabil, sedangkan Run ID #12 tetap menjadi konfigurasi dengan performa terbaik berdasarkan nilai F1-score.

4.3.7 Evaluasi Model

Tahap evaluasi mencakup pengukuran performa model menggunakan metrik Precision, Recall, dan F1-Score pada level entity. Evaluasi dilakukan pada test set yang tidak pernah dilihat model selama training.

4.3.7.1 Overall Metrics

Metrik keseluruhan (overall) mengukur performa model pada tingkat entitas (bukan per token) menggunakan pustaka seqeval. Precision adalah proporsi entitas yang diprediksi benar terhadap semua entitas yang diprediksi. Recall adalah proporsi entitas yang diprediksi benar terhadap semua entitas yang sebenarnya. F1-score adalah rata-rata harmonik dari precision dan recall. Metrik ini lebih informatif daripada akurasi token karena memperhitungkan batas entitas secara keseluruhan.

```
        # src/evaluator.py:76-78
        # Metrik overall: F1, Precision, Recall menggunakan seqeval
        overall_f1        = f1_score(all_trues_seq, all_preds_seq)
        overall_precision = precision_score(all_trues_seq, all_preds_seq)
overall_recall    = recall_score(all_trues_seq, all_preds_seq)
```

Metrik keseluruhan dihitung menggunakan fungsi f1_score, precision_score, dan recall_score dari pustaka seqeval. Fungsi-fungsi ini menerima dua argumen: daftar urutan label sebenarnya dan daftar urutan label prediksi. Setiap urutan mewakili satu dokumen atau satu chunk. Metrik dihitung pada tingkat entitas,

artinya suatu entitas dianggap benar hanya jika label dan batas karakternya tepat sama dengan ground truth.

4.3.7.2 Per-label Metrics

Selain metrik keseluruhan, penting untuk mengetahui performa model pada setiap jenis entitas (label). Metrik per-label ini menghitung Precision, Recall, dan F1-score untuk masing-masing entitas (misal: NOMOR_SURAT, TANGGAL, PENGIRIM, dan lain-lain) secara terpisah. Selain itu, nilai support menunjukkan berapa banyak kemunculan entitas tersebut dalam data uji, yang berguna untuk menilai apakah ketidakseimbangan kelas mempengaruhi performa.

```
    # src/evaluator.py:113-144
    # Per-label metrics: F1, Precision, Recall untuk setiap entitas
    def _parse_seqeval_report(trues, preds):
        result = {}
        for label in LABELS:
            # Filter prediksi yang relevan dengan label ini
            label_preds = [[t if t.endswith(label) else "O" for t in
seq] for seq in preds]
            label_trues = [[t if t.endswith(label) else "O" for t in
seq] for seq in trues]
            p = precision_score(label_trues, label_preds)
            r = recall_score(label_trues, label_preds)
            f = f1(label_trues, label_preds)
            # Support: jumlah entity label ini dalam data uji
            support = sum(1 for seq in label_trues for tag in seq if
tag == f"B-{label}")
            result[label] = {"precision": round(p, 4), "recall":
round(r, 4), "f1": round(f, 4), "support": support}
    return result
```

Fungsi parse_seqeval_report mengiterasi setiap label yang ada. Untuk setiap label, fungsi membuat salinan urutan prediksi dan ground truth dengan mengganti semua tag yang tidak sesuai label menjadi "O". Kemudian precision, recall, dan f1-score dari seqeval dihitung pada urutan yang telah difilter. Support dihitung dengan menghitung jumlah kemunculan tag B-label dalam data sebenarnya. Hasilnya disimpan dalam dictionary dengan kunci label.

4.3.7.3 Confusion Matrix

Confusion matrix (pada tingkat token) menunjukkan hubungan antara label sebenarnya dan label yang diprediksi oleh model. Matriks ini membantu

mengidentifikasi pasangan label yang sering tertukar. Misalnya, apakah model sering salah memprediksi I-LOKASI sebagai O atau sebagai I-PERIHAL. Analisis confusion matrix memberikan wawasan tentang kelemahan spesifik model sehingga dapat dilakukan perbaikan yang terarah.

```
    # src/evaluator.py:150-180
    # Confusion matrix token-level: true label vs predicted label
    def _build_confusion(trues, preds, top_n=10):
        matrix = defaultdict(lambda: defaultdict(int))
        for t, p in zip(trues, preds):
            matrix[t][p] += 1
        result = {}
        for true_lbl, pred_counts in sorted(matrix.items()):
            if true_lbl == "O":
                continue  # Skip label O untuk fokus pada entity
            correct = pred_counts.get(true_lbl, 0)
            total   = sum(pred_counts.values())
            result[true_lbl] = {
                "correct": correct, "total": total,
                "accuracy": round(correct / total, 4) if total else 0,
            }
    return result
```

Fungsi build_confusion membuat kamus yang memetakan label sebenarnya ke kamus lain yang memetakan label prediksi ke jumlah kemunculan. Fungsi mengiterasi pasangan label sebenarnya dan label prediksi untuk setiap token. Setelah semua token diproses, fungsi menyusun hasil untuk setiap label sebenarnya (kecuali label "O") dengan menghitung jumlah prediksi benar, total kemunculan label tersebut, dan akurasi per label. Hasil ini dapat divisualisasikan dalam bentuk tabel atau gambar confusion matrix.

4.3.7.4 Compute Metrics untuk Trainer

HuggingFace Trainer memerlukan fungsi compute_metrics yang akan dipanggil secara otomatis setelah setiap epoch evaluasi. Fungsi ini menerima tuple (logits, labels) dari hasil prediksi model, kemudian menghitung metrik Precision, Recall, dan F1-score pada tingkat entitas. Label dengan nilai -100 (yang menandakan token khusus atau padding) diabaikan. Hasil metrik digunakan oleh Trainer untuk memantau performa dan menyimpan checkpoint terbaik berdasarkan eval_f1.

```
    # src/finetune_indobert.py:298-330
    # Compute metrics function untuk HuggingFace Trainer
    def make_compute_metrics():
        def compute_metrics(eval_pred):
            logits, labels = eval_pred
```

```
            predictions = np.argmax(logits, axis=-1)
            true_seqs, pred_seqs = [], []
            for pred_row, label_row in zip(predictions, labels):
                t_seq, p_seq = [], []
                for p, l in zip(pred_row, label_row):
                    if l == -100:
                        continue  # Skip special tokens
                    t_seq.append(ID2LABEL.get(int(l), "O"))
                    p_seq.append(ID2LABEL.get(int(p), "O"))
                if t_seq:
                    true_seqs.append(t_seq)
                    pred_seqs.append(p_seq)
            return {
                "f1":        f1_score(true_seqs, pred_seqs),
                "precision": precision_score(true_seqs, pred_seqs),
                "recall":    recall_score(true_seqs, pred_seqs),
            }
    return compute_metrics
```

Fungsi make_compute_metrics mengembalikan fungsi dalam compute_metrics. Di dalamnya, logits diubah menjadi prediksi dengan argmax pada sumbu terakhir. Kemudian, untuk setiap baris prediksi dan label, fungsi menyaring token dengan label -100 (diabaikan). Label ID diubah menjadi nama label menggunakan pemetaan ID2LABEL. Urutan label sebenarnya dan prediksi dikumpulkan ke dalam daftar. Akhirnya, f1_score, precision_score, dan recall_score dari seqeval dihitung dan dikembalikan sebagai dictionary. Fungsi ini diatur sebagai compute_metrics pada Trainer.

4.3.7.5 Skor Kelengkapan Entitas

Selain metrik NER standar, sistem juga menghitung completeness score (skor kelengkapan) untuk mengevaluasi seberapa banyak jenis entitas yang berhasil diekstrak dari suatu dokumen. Skor ini dihitung sebagai proporsi label yang terisi (tidak None) terhadap total label yang didefinisikan (9 label). Skor 1,0 berarti semua entitas (NOMOR_SURAT, JENIS_DOKUMEN, TANGGAL, PENGIRIM, PENERIMA, PERIHAL, LOKASI, WAKTU, dan ISI) berhasil ditemukan, sementara skor 0,0 berarti tidak ada yang ditemukan. Metrik ini berguna untuk evaluasi kualitatif dari sudut pandang pengguna.

```
    # src/postprocess.py:281-301
    # Hitung skor kelengkapan entitas yang terisi
    def compute_completeness(entities):
        filled  = [lbl for lbl in LABELS if entities.get(lbl) is not
    None]
        missing = [lbl for lbl in LABELS if entities.get(lbl) is None]
        return {
```

```
            "score":    round(len(filled) / len(LABELS), 4),   # Contoh:
    7/9 = 0.7778
            "filled":   len(filled),
            "total":    len(LABELS),
            "missing":  missing,

    }
```

Fungsi compute_completeness menerima dictionary entities yang berisi hasil ekstraksi (kunci adalah label, nilai adalah teks entitas atau None). Fungsi membuat daftar filled berisi label-label yang nilainya bukan None. Daftar missing berisi label yang nilainya None. Skor kelengkapan dihitung sebagai panjang filled dibagi panjang total label. Fungsi mengembalikan dictionary dengan score (dibulatkan 4 desimal), filled (jumlah terisi), total (total label), dan missing (daftar label yang tidak terisi).

4.3.8 Implementasi Summarization

Tahap summarization mencakup pembuatan ringkasan naratif dalam Bahasa Indonesia berdasarkan entitas yang berhasil diekstrak. Proses ini menggunakan spaCy untuk tokenisasi kalimat dan rule-based generator untuk menghasilkan paragraf ringkasan yang mengalir secara alami.

4.3.8.1 Setup spaCy Tokenizer

spaCy digunakan sebagai pre-tokenizer untuk memisahkan teks menjadi kalimat (sentence splitting) dan kata-kata (word tokenization). Fungsi memuat model spaCy multibahasa ("xx_ent_wiki_sm") yang mendukung berbagai bahasa termasuk Indonesia. Jika model tidak tersedia (misal karena tidak diunduh), fungsi akan membuat model kosong berbahasa Indonesia dan menambahkan komponen sentencizer untuk pemisahan kalimat. Penggunaan spaCy memastikan tokenisasi yang konsisten dengan pipeline NER.

```
    # src/inference.py:122-135
    # Setup spaCy untuk tokenisasi kalimat dan kata
    def _get_nlp():
        global _nlp
        if _nlp is None:
            try:
                _nlp = spacy.load("xx_ent_wiki_sm")
            except OSError:
                _nlp = spacy.blank("id")
            if "sentencizer" not in _nlp.pipe_names:
                _nlp.add_pipe("sentencizer")
```

```
    return _nlp
```

Variabel global nlp digunakan untuk menyimpan objek spaCy agar tidak dimuat ulang setiap kali fungsi dipanggil. Fungsi mencoba memuat model xx_ent_wiki_sm (model multibahasa dengan entitas). Jika gagal (OSError), fungsi membuat model kosong untuk bahasa Indonesia. Kemudian, jika komponen sentencizer belum ada dalam pipeline, fungsi menambahkannya. Sentencizer adalah aturan sederhana untuk memisahkan kalimat berdasarkan tanda baca (titik, tanda seru, tanda tanya).

4.3.8.2 Paragraph Summary Generator

Fungsi generate_paragraph_summary menghasilkan ringkasan dokumen dalam bentuk satu paragraf narasi berbahasa Indonesia. Ringkasan dibangun dari entitas yang berhasil diekstrak oleh sistem. Pola kalimat disusun secara alami: dimulai dengan jenis dokumen (dan nomor serta tanggal jika ada), kemudian menyebutkan pengirim dan penerima, lalu perihal, dan diakhiri dengan informasi tambahan seperti lokasi atau waktu. Ringkasan dibatasi maksimal 6 kalimat untuk menjaga keringkasan.

```
# src/postprocess.py:355-468
# Buat ringkasan dokumen dalam 1 paragraf narasi Bahasa Indonesia
def generate_paragraph_summary(entities, filename=""):
    def _val(key, default=""):
        v = entities.get(key)
        if v is None: return default
        if isinstance(v, list): v = v[0] if v else default
        return str(v).strip()

    jenis     = _val("JENIS_DOKUMEN", "Dokumen")
    nomor     = _val("NOMOR_SURAT")
    tanggal   = _val("TANGGAL")
    pengirim  = _val("PENGIRIM")
    penerima  = _val("PENERIMA")
    perihal   = _val("PERIHAL")

    parts = []
    # Kalimat pembuka
    pembuka = jenis
    if nomor: pembuka += f" dengan nomor {nomor}"
    if tanggal: pembuka += f" tertanggal {tanggal}"
    parts.append(pembuka)

    # Pengirim & penerima
    if pengirim and penerima:
        parts.append(f"dikirimkan oleh {pengirim} kepada
```

```
        {penerima}")
            elif pengirim:
                parts.append(f"dikirimkan oleh {pengirim}")

            # Perihal
            if perihal:
                parts.append(f"mengenai {perihal}")

            # Contoh output:
            # "Surat keputusan dengan nomor 001/SK/2024 tertanggal 2024-
        01-12
            #  dikirimkan oleh Direktur kepada Kepala Divisi mengenai
            #  penetapan jabatan struktural..."

            sentences = []
            for part in parts:
                s = part[0].upper() + part[1:] + "."
                sentences.append(s)

    return " ".join(sentences[:6])
```

Fungsi mendefinisikan fungsi dalam _val untuk mengambil nilai entitas dengan aman: jika entitas tidak ada atau None, mengembalikan string kosong; jika nilai berupa list, mengambil elemen pertama. Selanjutnya, nilai-nilai entitas (JENIS_DOKUMEN, NOMOR_SURAT, TANGGAL, PENGIRIM, PENERIMA, PERIHAL) diambil. Bagian-bagian kalimat disusun dalam daftar parts: kalimat pembuka berisi jenis dokumen, nomor (jika ada), dan tanggal (jika ada); kalimat kedua (jika ada pengirim dan penerima) menyebutkan pengirim dan penerima; kalimat ketiga (jika ada perihal) menyebutkan perihal. Setiap bagian diubah menjadi kalimat dengan huruf awal kapital dan diakhiri titik. Hasil akhir adalah gabungan maksimal 6 kalimat pertama.

4.3.8.3 Normalisasi Entitas

Agar keluaran sistem konsisten dan mudah diproses lebih lanjut, nilai entitas perlu dinormalisasi. Normalisasi tanggal mengubah berbagai format (misal "12 Januari 2024" atau "12/01/2024") menjadi format ISO 8601 (YYYY-MM-DD). Normalisasi nomor surat membersihkan spasi berlebih di sekitar tanda garis miring (misal "001 / SK / 2024" menjadi "001/SK/2024"). Normalisasi juga dapat diterapkan pada entitas lain seperti nama orang (kapitalisasi konsisten) atau lokasi. Proses ini memastikan bahwa entitas yang sama memiliki representasi string yang identik, memudahkan pencarian dan pengelompokan.

```
    # src/postprocess.py:133-166
    # Normalisasi tanggal ke format ISO 8601 (YYYY-MM-DD)
    def normalize_tanggal(raw):
        # "12 Januari 2024" -> "2024-01-12"
        m      =      re.match(r"(\d{1,2})\s+(\w+)\s+(\d{4})",      raw,
    re.IGNORECASE)
        if m:
            day, month_str, year = m.groups()
            month = _BULAN.get(month_str.lower())
            if month:
                return f"{year}-{month}-{day.zfill(2)}"
        # "12/01/2024" -> "2024-01-12"
        m = re.match(r"(\d{1,2})[/\-](\d{1,2})[/\-](\d{4})", raw)
        if m:
            day, month, year = m.groups()
            return f"{year}-{month.zfill(2)}-{day.zfill(2)}"
        return raw


    # Normalisasi nomor surat (hapus spasi di sekitar slash)
    def normalize_nomor_surat(raw):
        # "001 / SK / 2024" -> "001/SK/2024"
    return re.sub(r"\s*/\s*", "/", raw).strip()
```

Fungsi normalize_tanggal mencoba beberapa pola regex: pertama untuk format "12 Januari 2024" (hari, bulan teks, tahun). Bulan teks dipetakan ke angka bulan melalui dictionary bulan. Hasil diformat sebagai "tahun-bulan-tanggal" dengan dua digit. Pola kedua untuk format "12/01/2024" atau "12-01-2024" (hari, bulan, tahun). Bulan dan hari diisi dengan dua digit. Jika tidak ada pola yang cocok, string asli dikembalikan. Fungsi normalize_nomor_surat mengganti spasi opsional di sekitar tanda slash dengan slash saja, lalu menghilangkan spasi di awal dan akhir.

4.3.8.4 Perhitungan Kelengkapan Entitas

Tahapan berikutnya setelah normalisasi adalah menghitung tingkat kelengkapan hasil ekstraksi entitas. Proses ini bertujuan untuk mengetahui seberapa banyak label yang berhasil dikenali oleh model dibandingkan dengan jumlah keseluruhan label yang digunakan pada sistem.

Perhitungan dilakukan dengan menghitung jumlah entitas yang memiliki nilai, kemudian membandingkannya dengan jumlah label yang tersedia. Selain menghasilkan nilai persentase (score), sistem juga mencatat label mana saja yang belum berhasil diekstraksi. Informasi ini selanjutnya disimpan ke dalam metadata keluaran JSON sehingga dapat digunakan untuk mengevaluasi kualitas hasil ekstraksi.

```
def compute_completeness(entities):
    from config import LABELS
    filled = [lbl for lbl in LABELS if entities.get(lbl) is not None]
    missing = [lbl for lbl in LABELS if entities.get(lbl) is None]
    return {
        "score": round(len(filled) / len(LABELS), 4),
        "filled": len(filled),
        "total": len(LABELS),
        "missing": missing,
    }
```

fungsi compute_completeness() digunakan untuk menghitung tingkat kelengkapan hasil ekstraksi entitas dengan membandingkan seluruh label yang telah didefinisikan pada sistem dengan hasil ekstraksi yang diperoleh dari model. Label yang memiliki nilai dimasukkan ke dalam daftar filled, sedangkan label yang belum memiliki nilai dimasukkan ke dalam daftar missing.

Selanjutnya fungsi menghitung nilai score sebagai rasio antara jumlah label yang berhasil diekstraksi terhadap jumlah keseluruhan label. Selain menghasilkan nilai persentase kelengkapan, fungsi juga mengembalikan jumlah label yang berhasil diekstraksi, jumlah keseluruhan label, serta daftar label yang belum berhasil dikenali. Informasi tersebut kemudian digunakan sebagai indikator kualitas hasil ekstraksi dan disimpan pada bagian metadata keluaran JSON.

4.3.8.5 Pembentukan Ringkasan Dokumen

Setelah seluruh entitas berhasil dinormalisasi dan divalidasi, sistem membentuk ringkasan dokumen dalam bentuk narasi. Ringkasan ini disusun secara otomatis menggunakan informasi hasil ekstraksi, seperti jenis dokumen, nomor surat, tanggal, pengirim, penerima, perihal, isi dokumen, lokasi, serta informasi tabel apabila tersedia.

Penyusunan ringkasan dilakukan secara bertahap dengan menggabungkan setiap entitas ke dalam beberapa kalimat sehingga menghasilkan paragraf yang mudah dipahami oleh pengguna. Apabila suatu entitas tidak tersedia, sistem akan

melewati bagian tersebut sehingga ringkasan tetap terbentuk secara alami tanpa menampilkan nilai yang kosong.

```
        # src/postprocess.py:133-166
jenis = _val("JENIS_DOKUMEN", "Dokumen")

nomor = _val("NOMOR_SURAT")

tanggal = _val("TANGGAL")

pembuka = jenis

if nomor:

    pembuka += f" dengan nomor {nomor}"

if tanggal:

    pembuka += f" tertanggal {tanggal}"

parts.append(pembuka)
```

fungsi generate_paragraph_summary() diawali dengan mengambil nilai setiap entitas menggunakan fungsi _val(). Informasi mengenai jenis dokumen, nomor surat, dan tanggal kemudian digunakan untuk menyusun kalimat pembuka sebagai dasar ringkasan dokumen.

Selanjutnya fungsi menambahkan informasi lain, seperti pengirim, penerima, perihal, isi dokumen, lokasi, serta tabel apabila tersedia. Setiap informasi hanya ditambahkan jika nilai entitas berhasil diperoleh sehingga paragraf yang dihasilkan tetap mengalir secara alami. Hasil akhir dari proses ini berupa ringkasan dokumen dalam satu paragraf yang akan disertakan pada keluaran sistem.

4.3.8.6 Normalisasi Entitas

Tahap terakhir pada proses post-processing adalah membangun struktur keluaran dalam format JSON. Pada tahap ini seluruh hasil ekstraksi yang telah melalui proses validasi, normalisasi, serta penyempurnaan digabungkan ke dalam satu objek JSON yang siap digunakan oleh layanan REST API.

Selain menyimpan hasil ekstraksi entitas, sistem juga menambahkan metadata dokumen yang meliputi nama file, jenis PDF, jumlah halaman, nilai OCR confidence, waktu pemrosesan, serta informasi tingkat kelengkapan hasil ekstraksi.

Ringkasan dokumen yang telah dibentuk pada tahap sebelumnya juga dimasukkan ke dalam struktur keluaran JSON.

```
output = {
    "metadata": {
        "sumber_file": filename,
        "jenis_pdf": pdf_type,
        "jumlah_halaman": page_count,
        "ocr_confidence": ocr_confidence,
        "kelengkapan": completeness,
    },
    "ringkasan": {
        "NOMOR_SURAT": normalized.get("NOMOR_SURAT"),
        "JENIS_DOKUMEN": normalized.get("JENIS_DOKUMEN"),
        "TANGGAL": normalized.get("TANGGAL"),
        "PENGIRIM": normalized.get("PENGIRIM"),
        "PENERIMA": normalized.get("PENERIMA"),
        "PERIHAL": normalized.get("PERIHAL"),
    },
    "paragraph_summary": paragraph,
}
```

fungsi build_output_json() menyusun seluruh hasil ekstraksi ke dalam struktur JSON yang terdiri atas tiga bagian utama, yaitu metadata, ringkasan, dan paragraph_summary. Bagian metadata menyimpan informasi mengenai dokumen yang diproses beserta tingkat kelengkapan hasil ekstraksi, sedangkan bagian ringkasan berisi nilai setiap entitas yang telah melalui proses validasi dan normalisasi.

Selain itu, fungsi juga menyertakan paragraph_summary yang berisi ringkasan dokumen dalam bentuk narasi sehingga pengguna dapat memahami isi dokumen secara cepat tanpa harus membaca keseluruhan dokumen. Struktur JSON yang dihasilkan merupakan keluaran akhir dari proses post-processing dan selanjutnya

dikirimkan melalui layanan REST API sebagai respons kepada pengguna.

4.3.9 Implementasi REST API

Sistem diimplementasikan menggunakan arsitektur REST API berbasis framework FastAPI. Framework ini dipilih karena memiliki performa tinggi, mendukung asynchronous processing, serta mampu menghasilkan dokumentasi API secara otomatis melalui Swagger UI. Dengan pendekatan ini, model dapat diakses oleh berbagai aplikasi tanpa harus melakukan integrasi langsung terhadap kode sumber model.

Implementasi API mencakup beberapa endpoint utama yang digunakan untuk mengelola proses analisis dokumen. Endpoint tersebut meliputi layanan analisis dokumen, analisis teks langsung, informasi label entitas, monitoring layanan, serta pemrosesan asynchronous. Setiap permintaan yang diterima akan melalui proses validasi untuk memastikan format data yang dikirim sesuai dengan kebutuhan sistem.

Setelah dokumen diterima, API akan meneruskan data ke pipeline pemrosesan yang terdiri dari ekstraksi teks, preprocessing, inferensi model NER, ekstraksi tabel, dan pembentukan ringkasan. Hasil yang diperoleh kemudian dikembalikan kepada pengguna dalam format JSON. Pendekatan ini memungkinkan sistem digunakan sebagai layanan yang dapat diintegrasikan dengan aplikasi web, aplikasi mobile, maupun sistem informasi lainnya.

4.3.10 Implementasi Deployment

Tahap berikutnya adalah deployment ke lingkungan produksi menggunakan layanan AWS EC2. Penggunaan AWS EC2 dipilih karena menyediakan fleksibilitas konfigurasi server serta kemampuan untuk menjalankan model machine learning secara mandiri tanpa ketergantungan pada layanan pihak ketiga. Dengan deployment ini, sistem dapat diakses melalui jaringan internet dan digunakan secara real-time.

Proses deployment dimulai dengan menyiapkan instance AWS EC2 yang telah dikonfigurasi dengan sistem operasi Linux. Seluruh dependensi yang dibutuhkan oleh aplikasi, termasuk Python, FastAPI, PyTorch, PaddleOCR, dan model hasil fine-tuning, diinstal pada server. Setelah itu aplikasi dijalankan menggunakan Uvicorn sebagai ASGI server yang bertugas menangani permintaan dari pengguna.

Untuk meningkatkan stabilitas layanan, digunakan Nginx sebagai reverse proxy yang meneruskan permintaan pengguna ke aplikasi FastAPI yang berjalan pada server. Selain itu dilakukan konfigurasi keamanan seperti pengaturan firewall, manajemen port, dan pembatasan akses server. Dengan konfigurasi tersebut, sistem mampu menyediakan layanan analisis dokumen administratif secara stabil, aman, dan dapat diakses oleh pengguna melalui endpoint API yang telah disediakan.

4.4 Pengujian

Pengujian sistem dilakukan untuk memvalidasi bahwa sistem analisis dokumen administratif yang dikembangkan telah memenuhi kebutuhan yang ditetapkan serta mampu menjalankan seluruh fungsi sesuai dengan tujuan penelitian. Pengujian dilakukan terhadap keseluruhan komponen sistem yang meliputi proses unggah dokumen, ekstraksi teks menggunakan OCR, identifikasi entitas menggunakan Named Entity Recognition (NER), pembentukan ringkasan dokumen, hingga penyajian hasil melalui layanan REST API.

Pengujian menggunakan pendekatan berlapis yang terdiri atas tiga metode utama, yaitu pengujian fungsional (black box testing), evaluasi performa model Named Entity Recognition (NER), dan User Acceptance Testing (UAT). Pengujian fungsional digunakan untuk memastikan bahwa seluruh fitur sistem berjalan sesuai dengan kebutuhan fungsional yang telah ditetapkan. Evaluasi model dilakukan untuk mengukur kemampuan model dalam mengenali entitas pada dokumen administratif menggunakan metrik Precision, Recall, F1-Score, dan Confusion Matrix. Sementara itu, User Acceptance Testing dilakukan untuk mengetahui tingkat penerimaan pengguna terhadap sistem yang dikembangkan berdasarkan skenario penggunaan nyata.

4.4.1 Deskripsi Pengujian

1. Pengujian Fungsional *Black Box*

Pengujian fungsional dilakukan menggunakan metode black box testing, yaitu pendekatan pengujian yang berfokus pada perilaku sistem berdasarkan masukan dan keluaran yang dihasilkan tanpa memperhatikan implementasi kode program di dalam sistem. Pengujian dilakukan terhadap seluruh fitur utama sistem yang telah dirancang pada tahap analisis kebutuhan. Fitur yang diuji meliputi proses autentikasi pengguna, unggah dokumen, ekstraksi teks, preprocessing, identifikasi entitas menggunakan model NER, pembentukan ringkasan dokumen, serta layanan REST API.

2. Evaluasi Model Named Entity Recognition (NER)

Evaluasi model dilakukan untuk mengukur kemampuan model IndoBERT hasil fine-tuning dalam mengenali entitas administratif yang terdapat pada dokumen. Pengujian dilakukan menggunakan data pengujian yang tidak pernah digunakan selama proses pelatihan maupun validasi. Evaluasi dilakukan menggunakan metrik Precision, Recall, F1-Score, dan analisis Confusion Matrix. Selain itu dilakukan analisis terhadap label yang masih sering mengalami kesalahan prediksi untuk mengetahui keterbatasan model dalam melakukan generalisasi terhadap berbagai jenis dokumen.

3. Prosedur *User Acceptance Testing*

User Acceptance Testing (UAT) dilakukan untuk mengetahui tingkat penerimaan pengguna terhadap sistem yang dikembangkan. Pengujian ini berfokus pada kesesuaian sistem dengan kebutuhan pengguna serta kemudahan penggunaan sistem dalam mendukung proses analisis dokumen administratif.

Pengujian dilakukan menggunakan berbagai jenis dokumen administratif yang umum digunakan dalam lingkungan institusi, yaitu Nota Dinas, Surat Keterangan Lulus, Surat Keterangan, Surat Permohonan, Surat Undangan, Pengumuman, dan Pengalihan Perkuliahan. Selain itu, beberapa dokumen dengan struktur yang lebih kompleks seperti TOR, Lampiran, dan Transkrip juga digunakan untuk menguji

kemampuan sistem dalam melakukan generalisasi terhadap dokumen yang memiliki format berbeda.

Penilaian dilakukan berdasarkan keberhasilan sistem dalam menghasilkan ringkasan dokumen yang sesuai dengan informasi yang terdapat pada dokumen asli. Hasil pengujian kemudian digunakan untuk mengetahui tingkat kepuasan pengguna serta mengidentifikasi area yang masih memerlukan perbaikan pada pengembangan selanjutnya.

4.4.2 Data Hasil Pengujian

1. Hasil Pengujian Fungsional *Black Box*

Pengujian fungsional dilakukan terhadap 21 skenario pengujian yang dikelompokkan berdasarkan modul sistem. Tabel 11 menyajikan hasil lengkap pengujian tersebut.

Tabel 4.5 Tabel Hasil Pengujian Functional Blackbox

ID	Modul	Skenario	Masukan	Keluaran yang Diharapkan	Status
TC-DOC-01	Upload Dokumen	Upload dokumen PDF valid	File PDF dokumen administratif	Dokumen berhasil diunggah dan tersimpan	Valid
TC-DOC-02	Upload Dokumen	Upload dokumen DOCX valid	File DOCX dokumen administratif	Dokumen berhasil diunggah dan tersimpan	Valid
TC-DOC-03	Upload Dokumen	Upload file tidak didukung	File JPG/PNG	Sistem menampilkan pesan format tidak didukung	Valid

ID	Modul	Skenario	Masukan	Keluaran yang Diharapkan	Status
TC-PROC-01	Preprocessing	Menjalankan preprocessing dokumen	Dokumen yang telah diunggah	Sistem melakukan pembersihan dan normalisasi teks	Valid
TC-OCR-01	OCR	Ekstraksi teks dari dokumen PDF	Dokumen PDF hasil scan	System berhasil mengekstak teks	Valid
TC-NER-01	NER	Identifikasi entitas dokumen	Teks hasil OCR	Sistem mengenali entitas NOMOR_SURAT, TANGGAL, PENGIRIM, PENERIMA, PERIHAL, dan LOKASI	Valid
TC-NER-02	NER	Identifikasi entitas pada dokumen Nota Dinas	Dokumen Nota Dinas	Sistem menghasilkan label entitas yang sesuai	Valid
TC-NER-03	NER	Identifikasi entitas pada Surat Keterangan Lulus	Dokumen Surat Keterangan Lulus	Sistem menghasilkan label entitas yang sesuai	Valid

ID	Modul	Skenario	Masukan	Keluaran yang Diharapkan	Status
TC-NER-04	NER	Identifikasi entitas pada Surat Keterangan	Dokumen Surat Keterangan	Sistem menghasilkan label entitas yang sesuai	Valid
TC-NER-05	NER	Identifikasi entitas pada Surat Permohonan	Dokumen Surat Permohonan	Sistem menghasilkan label entitas yang sesuai	Valid
TC-NER-06	NER	Identifikasi entitas pada Surat Undangan	Dokumen Surat Undangan	Sistem menghasilkan label entitas yang sesuai	Valid
TC-NER-07	NER	Identifikasi entitas pada Pengumuman	Dokumen Pengumuman	Sistem menghasilkan label entitas yang sesuai	Valid
TC-NER-08	NER	Identifikasi entitas pada Pengalihan Perkuliahan	Dokumen Pengalihan Perkuliahan	Sistem menghasilkan label entitas yang sesuai	Valid
TC-NER-09	NER	Identifikasi entitas pada dokumen TOR	Dokumen TOR	Sebagian besar entitas berhasil dikenali, terdapat missing label	Valid dengan Catatan

ID	Modul	Skenario	Masukan	Keluaran yang Diharapkan	Status
				pada beberapa bagian	
TC-NER-10	NER	Identifikasi entitas pada dokumen Lampiran	Dokumen Lampiran	Sebagian dikenali terdapat missing label pada beberapa bagian	Valid dengan Catatan
TC-NER-11	NER	Identifikasi entitas pada dokumen Transkrip	Dokumen Transkrip	Sebagian besar entitas berhasil dikenali, terdapat missing label pada beberapa bagian	Valid dengan Catatan
TC-NER-12	NER	Identifikasi entitas pada dokumen Non Jurusan TIK	Dokumen Non Jurusan TIK	Sebagian besar entitas tidak di kenali, label yang tampil menjadi null	Valid
TC-SUM-01	Ringkasan Dokumen	Menekan tombol Generate Summary	Dokumen yang telah diproses	Sistem menghasilkan ringkasan dokumen otomatis	Valid
TC-SUM-	Ringkasan Dokumen	Ringkasan dokumen Nota	Dokumen Nota Dinas	Ringkasan sesuai dengan isi dokumen	Valid

ID	Modul	Skenario	Masukan	Keluaran yang Diharapkan	Status
02		Dinas			
TC-SUM-03	Ringkasan Dokumen	Ringkasan dokumen Surat Undangan	Dokumen Surat Undangan	Ringkasan sesuai dengan isi dokumen	Valid
TC-API-01	REST API	Permintaan analisis dokumen	Dokumen administratif	API mengembalikan hasil analisis dalam format JSON	Valid
TC-API-02	REST API	Permintaan hasil ringkasan	Dokumen yang telah diproses	API mengembalikan ringkasan dokumen dalam format JSON	Valid

Seluruh 22 skenario pengujian fungsional menghasilkan keluaran yang sesuai dengan spesifikasi yang diharapkan, dengan tingkat keberhasilan 100% (22/22).

2. Hasil Evaluasi Model NER

Evaluasi model menunjukkan bahwa model IndoBERT yang telah melalui proses fine-tuning mampu mengenali sebagian besar entitas administratif dengan baik. Berdasarkan hasil pengujian diperoleh nilai Precision sebesar 0,8103, Recall sebesar 0,8868, F1-Score sebesar 0,8468, dan loss sebesar 0,0705.

Tabel 4.6 Hasil Evaluasi Model

Metrik	Nilai

Metrik	Nilai
Precision	0.8103
Recall	0.8868
F1-Score	0,8468
Loss	0.705

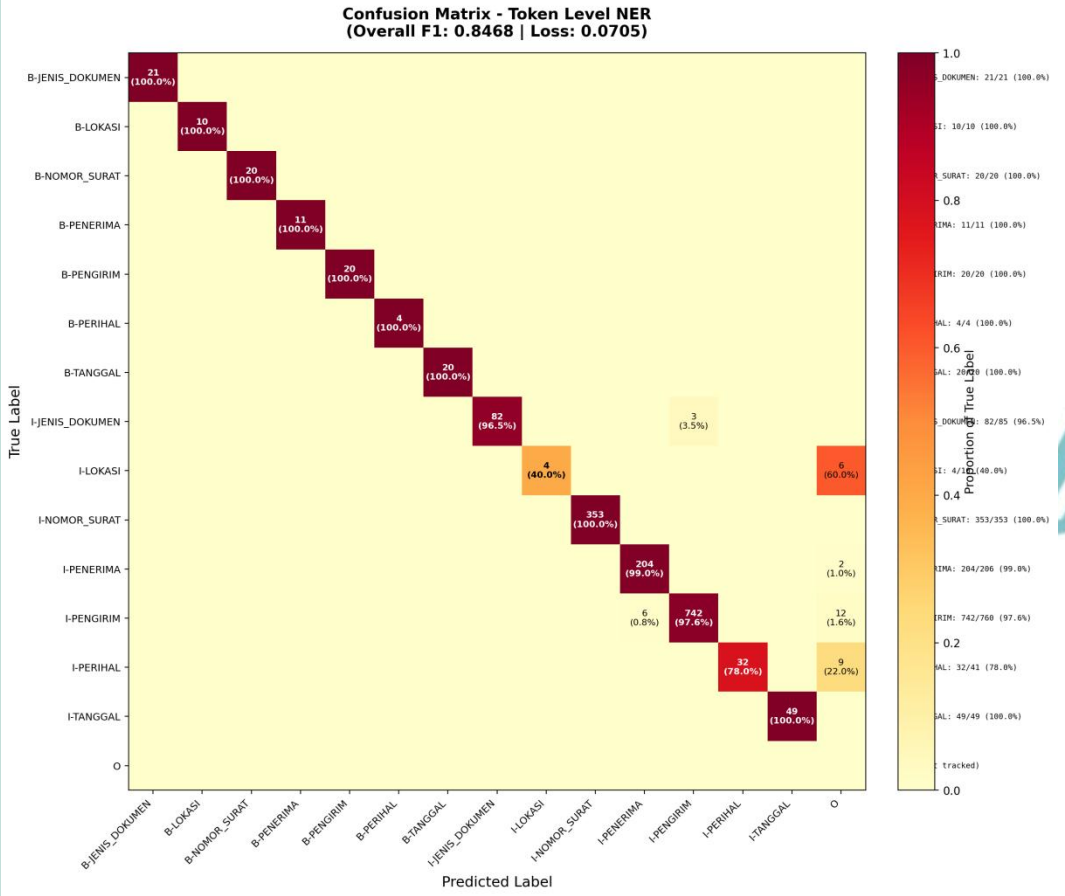
Berdasarkan hasil tersebut, dapat disimpulkan bahwa model memiliki kemampuan yang baik dalam mengenali informasi penting pada dokumen administratif. Nilai Recall yang lebih tinggi dibandingkan Precision menunjukkan bahwa model cenderung mampu menemukan sebagian besar entitas yang terdapat pada dokumen. Namun demikian, masih terdapat beberapa prediksi yang kurang tepat sehingga menyebabkan nilai Precision sedikit lebih rendah dibandingkan Recall. Kondisi ini menunjukkan bahwa model relatif lebih baik dalam menemukan entitas daripada menghindari kesalahan klasifikasi.

Untuk memperoleh gambaran yang lebih rinci mengenai performa model pada setiap kategori entitas, dilakukan analisis menggunakan confusion matrix. Analisis ini digunakan untuk mengetahui distribusi prediksi yang dihasilkan model pada masing-masing label sehingga dapat diketahui kategori entitas yang memiliki performa tinggi maupun kategori yang masih sering mengalami kesalahan prediksi.

Tabel 4.7 Tabel Confusion Matriks Per Label

Label	Akurasi
B-JENIS_DOKUMEN	100%
B-LOKASI	100%
B-NOMOR_SURAT	100%
B-PENERIMA	100%
B-PENGIRIM	100%
B-PERIHAL	100%
B-TANGGAL	100%
I-JENIS_DOKUMEN	96.5%
I-LOKASI	40%
I-NOMOR_SURAT	100%

Label	Akurasi
I-PENERIMA	99%
I-PENGIRIM	97.6%
I-PERIHAL	78%
I-TANGGAL	100%



Gambar 4.6 Hasil Confusion Matriks

Berdasarkan confusion matrix yang diperoleh, seluruh label dengan awalan B- (Beginning) berhasil dikenali dengan sangat baik oleh model dengan tingkat akurasi mencapai 100%. Hasil ini menunjukkan bahwa model mampu menentukan awal suatu entitas secara konsisten pada seluruh kategori yang digunakan dalam penelitian, yaitu JENIS_DOKUMEN, NOMOR_SURAT, TANGGAL, PENGIRIM, PENERIMA, PERIHAL, dan LOKASI.

Pada kategori I- (Inside), sebagian besar label juga menunjukkan performa yang sangat baik. Label I-NOMOR_SURAT dan I-TANGGAL memperoleh akurasi sebesar 100%, yang menunjukkan bahwa model mampu mengenali keseluruhan isi entitas secara konsisten tanpa mengalami kesalahan prediksi. Label I-PENERIMA dan I-PENGIRIM juga menunjukkan performa yang tinggi dengan akurasi masing-masing sebesar 99,0% dan 97,6%. Hasil ini mengindikasikan bahwa informasi terkait pihak pengirim dan penerima surat memiliki pola penulisan yang relatif konsisten sehingga lebih mudah dipelajari oleh model selama proses pelatihan.

Label I-JENIS_DOKUMEN memperoleh akurasi sebesar 96,5%, yang menunjukkan bahwa model mampu mengenali sebagian besar jenis dokumen administratif dengan baik. Kesalahan yang terjadi pada kategori ini relatif kecil dan umumnya disebabkan oleh variasi penulisan nama dokumen yang tidak banyak muncul pada data pelatihan.

Sementara itu, performa yang lebih rendah ditemukan pada label I-PERIHAL dan I-LOKASI. Label I-PERIHAL memperoleh akurasi sebesar 78,0%, sedangkan label I-LOKASI hanya memperoleh akurasi sebesar 40,0%. Hasil ini menunjukkan bahwa kedua kategori tersebut merupakan entitas yang paling sulit dikenali oleh model dibandingkan kategori lainnya.

Kesalahan prediksi pada label I-PERIHAL umumnya terjadi karena bagian perihal memiliki panjang teks yang bervariasi dan sering mengandung kata-kata yang juga muncul pada kategori lain. Variasi konteks tersebut menyebabkan model mengalami kesulitan dalam menentukan batas akhir entitas secara konsisten.

Akibatnya, beberapa token yang seharusnya diberi label I-PERIHAL diprediksi sebagai label O (Outside).

Pada label I-LOKASI, tingkat akurasi yang lebih rendah disebabkan oleh tingginya variasi format penulisan lokasi pada dokumen administratif. Beberapa lokasi ditulis dalam bentuk nama gedung, ruang, alamat lengkap, maupun kombinasi beberapa lokasi sekaligus. Variasi ini menyebabkan jumlah pola yang harus dipelajari model menjadi lebih banyak sehingga meningkatkan kemungkinan terjadinya kesalahan klasifikasi. Selain itu, jumlah sampel lokasi pada dataset juga relatif lebih sedikit dibandingkan kategori lainnya sehingga model memiliki keterbatasan dalam mempelajari karakteristik entitas tersebut.

Temuan pada confusion matrix ini sejalan dengan hasil pengujian User Acceptance Testing yang menunjukkan bahwa beberapa dokumen dengan struktur yang lebih kompleks, seperti TOR, Lampiran, dan Transkrip, masih menghasilkan kasus missing label pada entitas tertentu. Sebagian besar kasus tersebut berkaitan dengan entitas lokasi dan perihal yang memiliki variasi konteks lebih tinggi dibandingkan entitas lainnya.

Secara keseluruhan, hasil evaluasi menunjukkan bahwa model IndoBERT hasil fine-tuning telah mampu melakukan identifikasi entitas administratif dengan performa yang baik. Nilai F1-Score sebesar 0,8468 menunjukkan bahwa pendekatan yang digunakan pada penelitian ini efektif untuk mendukung proses identifikasi bagian teks variabel dinamis pada dokumen administratif. Meskipun masih terdapat beberapa keterbatasan pada kategori tertentu, performa yang diperoleh sudah cukup untuk mendukung implementasi sistem analisis dokumen administratif secara otomatis sesuai dengan tujuan penelitian.

3. Hasil evaluasi BERTscore dan ROUGE

Evaluasi dilakukan untuk mengukur tingkat kesesuaian antara hasil ringkasan yang dihasilkan oleh sistem dengan ground truth dengan 25 sample. Pada penelitian ini digunakan dua metode evaluasi, yaitu ROUGE (Recall-Oriented Understudy for Gisting Evaluation) dan BERTScore. ROUGE digunakan untuk mengukur tingkat kesamaan secara leksikal berdasarkan kecocokan kata dan struktur kalimat, sedangkan BERTScore digunakan untuk mengukur kesamaan

makna (semantic similarity) menggunakan representasi embedding dari model BERT. Dengan mengombinasikan kedua metode tersebut, evaluasi dapat memberikan gambaran yang lebih komprehensif mengenai kualitas hasil ringkasan, baik dari sisi kesamaan kata maupun kesamaan makna terhadap ground truth.

Hasil evaluasi rata-rata terhadap seluruh dokumen pengujian ditunjukkan pada Tabel 4.8.

Tabel 4.8 Hasil Evaluasi Menggunakan ROUGE dan BERTScore

Metode Evaluasi	Precision	Recall	F1-score
ROUGE-1	0.8078	0.3827	0.4981
ROUGE-2	0.6121	0.3019	0.3886
ROUGE-L	0.5558	0.2589	0.3379
BERTscore	0.7960	0.7109	.07501

ROUGE-1 memperoleh nilai Precision sebesar 0,8078, Recall sebesar 0,3827, dan F1-Score sebesar 0,4981. Nilai tersebut menunjukkan bahwa sistem mampu mempertahankan sebagian besar kata-kata penting dari ground truth, meskipun tidak seluruh informasi berhasil diekstraksi sehingga nilai Recall relatif lebih rendah.

Pada ROUGE-2, diperoleh Precision sebesar 0,6121, Recall sebesar 0,3019, dan F1-Score sebesar 0,3886. Nilai yang lebih rendah dibandingkan ROUGE-1 menunjukkan bahwa sistem tidak selalu mempertahankan urutan pasangan kata (bigram) yang sama dengan ground truth, akibat proses ekstraksi yang hanya mengambil informasi penting.

Sementara itu, ROUGE-L menghasilkan Precision sebesar 0,5558, Recall sebesar 0,2589, dan F1-Score sebesar 0,3379. Hasil ini menunjukkan bahwa struktur kalimat hasil ekstraksi memiliki perbedaan dengan ground truth, meskipun informasi utama masih dapat dipertahankan.

Berbeda dengan ROUGE, BERTScore memperoleh Precision sebesar 0,7960, Recall sebesar 0,7109, dan F1-Score sebesar 0,7501. Nilai tersebut menunjukkan bahwa hasil ekstraksi memiliki tingkat kesamaan makna yang tinggi terhadap ground truth, walaupun terdapat perbedaan pada pemilihan kata maupun struktur kalimat.

Secara keseluruhan, hasil evaluasi menunjukkan bahwa sistem lebih baik dalam mempertahankan makna informasi dibandingkan mempertahankan kesamaan kata secara langsung. Hal ini ditunjukkan oleh nilai BERTScore F1 sebesar 0,7501, yang lebih tinggi dibandingkan seluruh metrik ROUGE. Dengan demikian, sistem mampu mengekstraksi informasi penting dari dokumen administratif dengan baik meskipun terjadi perubahan pada bentuk atau susunan kalimat hasil ekstraksi.

4. Hasil *User Acceptance Testing*

User Acceptance Testing (UAT) dilakukan untuk mengevaluasi tingkat penerimaan pengguna terhadap sistem analisis dokumen administratif yang dikembangkan. Berbeda dengan pengujian fungsional yang berfokus pada validasi fitur sistem, UAT bertujuan untuk menilai apakah sistem telah memenuhi kebutuhan pengguna dalam kondisi penggunaan yang mendekati lingkungan operasional sebenarnya.

Pengujian dilakukan dengan menjalankan serangkaian skenario yang mewakili alur penggunaan sistem secara lengkap, mulai dari proses autentikasi pengguna, unggah dokumen, proses analisis dokumen, hingga pemeriksaan hasil ringkasan yang dihasilkan. Pengujian menggunakan berbagai jenis dokumen administratif yang menjadi objek penelitian, yaitu Nota Dinas, Surat Keterangan Lulus, Surat Keterangan, Surat Permohonan, Surat Undangan, Pengumuman, dan Pengalihan Perkuliahan. Selain itu, pengujian juga dilakukan pada dokumen TOR, Lampiran, dan Transkrip untuk mengukur kemampuan sistem dalam melakukan generalisasi terhadap dokumen yang memiliki struktur berbeda dari data pelatihan.

Selama proses pengujian, pengguna diminta untuk membandingkan hasil ringkasan yang dihasilkan sistem dengan isi dokumen asli. Hasil pengujian kemudian dicatat berdasarkan tingkat kesesuaian antara keluaran sistem dan informasi yang terdapat pada dokumen sumber.

Tabel 4.9 Tabel Skenario dan Hasil UAT

Kode KF	Skenario *UAT*	Hasil yang Diharapkan	Status
KF-01	Menguji proses masuk ke sistem menggunakan akun terdaftar	Pengguna berhasil masuk dan diarahkan ke halaman utama	Sesuai
KF-02	Menguji proses unggah dokumen administratif	Dokumen berhasil diunggah dan tersimpan pada sistem	Sesuai
KF-03	Menguji proses ekstraksi teks menggunakan OCR	Teks pada dokumen berhasil diekstraksi dan dapat diproses lebih lanjut	Sesuai
KF-04	Menguji proses identifikasi entitas menggunakan model NER	Sistem berhasil mengenali entitas seperti nomor surat, tanggal, pengirim, penerima, perihal, lokasi, dan jenis dokumen	Sesuai
KF-05	Menguji proses pembentukan ringkasan dokumen menggunakan	Sistem menghasilkan ringkasan dokumen secara otomatis	Sesuai

Kode KF	Skenario *UAT*	Hasil yang Diharapkan	Status
	tombol Generate Summary		
KF-06	Menguji analisis dokumen Nota Dinas	Sistem menghasilkan ringkasan yang sesuai dengan isi dokumen Nota Dinas	Sesuai
KF-07	Menguji analisis dokumen Surat Keterangan Lulus	Sistem menghasilkan ringkasan yang sesuai dengan isi dokumen Surat Keterangan Lulus	Sesuai
KF-08	Menguji analisis dokumen Surat Keterangan	Sistem menghasilkan ringkasan yang sesuai dengan isi dokumen Surat Keterangan	Sesuai
KF-09	Menguji analisis dokumen Surat Permohonan	Sistem menghasilkan ringkasan yang sesuai dengan isi dokumen Surat Permohonan	Sesuai
KF-10	Menguji analisis dokumen Surat	Sistem menghasilkan ringkasan yang sesuai dengan isi dokumen Surat	Sesuai

Kode KF	Skenario *UAT*	Hasil yang Diharapkan	Status
	Undangan	Undangan	
KF-11	Menguji analisis dokumen Pengumuman	Sistem menghasilkan ringkasan yang sesuai dengan isi dokumen Pengumuman	Sesuai
KF-12	Menguji analisis dokumen Pengalihan Perkuliahan	Sistem menghasilkan ringkasan yang sesuai dengan isi dokumen Pengalihan Perkuliahan	Sesuai
KF-13	Menguji analisis dokumen TOR	Sistem mampu menghasilkan ringkasan dokumen, namun masih ditemukan beberapa missing label pada entitas tertentu	Sesuai Dengan Catatan
KF-14	Menguji analisis dokumen Lampiran	Sistem mampu menghasilkan ringkasan dokumen, namun masih ditemukan beberapa missing label pada entitas tertentu	Sesuai Dengan Catatan

Kode KF	Skenario *UAT*	Hasil yang Diharapkan	Status
KF-15	Menguji analisis dokumen Transkrip	Sistem mampu menghasilkan ringkasan dokumen, namun masih ditemukan beberapa missing label pada entitas tertentu	Sesuai Dengan Catatan
KF-16	Menguji keluaran hasil dalam format JSON	Sistem menghasilkan keluaran JSON yang valid dan sesuai dengan hasil analisis	Sesuai
KF-17	Menguji analisis beberapa dokumen secara berulang	Sistem tetap berjalan stabil dan menghasilkan keluaran yang benar	Sesuai

seluruh kebutuhan fungsional sistem dapat dijalankan sesuai dengan harapan pengguna. Sebanyak 14 skenario memperoleh status Sesuai, sedangkan 3 skenario memperoleh status Sesuai dengan Catatan, yaitu pada pengujian dokumen TOR, Lampiran, dan Transkrip. Pada ketiga jenis dokumen tersebut sistem tetap mampu menghasilkan ringkasan dan keluaran JSON, namun masih ditemukan beberapa kasus missing label pada entitas tertentu. Meskipun demikian, tidak ditemukan skenario dengan status Tidak Sesuai, sehingga dapat disimpulkan bahwa sistem telah memenuhi kebutuhan pengguna dan layak digunakan untuk membantu proses analisis dokumen administratif secara otomatis.

Tabel 4.10 Tabel Tanggapan Terbuka Peserta UAT

Aspek	Tanggapan Peserta
Fitur yang paling disukai	Peserta menyatakan bahwa fitur yang paling membantu adalah kemampuan sistem dalam melakukan identifikasi informasi penting secara otomatis dari dokumen administratif serta menghasilkan ringkasan dokumen tanpa perlu membaca keseluruhan isi dokumen. Selain itu, proses analisis yang dilakukan melalui satu tombol Generate Summary dinilai mempermudah pekerjaan administrasi.
Masukan perbaikan 1	Sistem perlu meningkatkan kemampuan identifikasi entitas pada dokumen yang memiliki struktur kompleks, terutama dokumen TOR, Lampiran, dan Transkrip yang masih menghasilkan beberapa kasus missing label.
Masukan perbaikan 2	Perlu dilakukan penambahan variasi data pelatihan agar model mampu mengenali lebih banyak pola penulisan dokumen administratif yang berbeda-beda.
Masukan perbaikan 3	Perlu dilakukan optimasi proses OCR untuk meningkatkan kualitas ekstraksi teks pada dokumen hasil pemindaian yang memiliki kualitas gambar rendah atau tata letak yang tidak terstruktur.
Masukan pengembangan lanjutan	Sistem dapat dikembangkan lebih lanjut dengan menambahkan dukungan jenis dokumen yang lebih beragam, meningkatkan akurasi model NER, serta mengintegrasikan layanan API dengan sistem administrasi institusi sehingga proses analisis dokumen dapat dilakukan secara otomatis pada skala yang lebih besar.

Berdasarkan hasil User Acceptance Testing, pengguna memberikan tanggapan yang positif terhadap sistem yang dikembangkan. Fitur yang paling diapresiasi adalah kemampuan sistem dalam melakukan analisis dokumen secara otomatis dan menghasilkan ringkasan informasi penting dalam waktu yang relatif singkat. Pengguna menilai bahwa sistem dapat membantu mengurangi waktu yang diperlukan untuk membaca dan memahami dokumen administratif yang memiliki jumlah halaman cukup banyak.

Meskipun demikian, pengguna juga memberikan beberapa masukan untuk pengembangan sistem pada penelitian selanjutnya. Masukan utama berkaitan dengan peningkatan kemampuan model dalam mengenali entitas pada dokumen yang memiliki struktur lebih kompleks, seperti TOR, Lampiran, dan Transkrip. Selain itu, pengguna menyarankan penambahan variasi dataset pelatihan serta optimasi proses OCR agar kualitas ekstraksi teks dapat ditingkatkan. Secara keseluruhan, hasil UAT menunjukkan bahwa sistem telah diterima dengan baik oleh pengguna dan dinilai mampu membantu proses identifikasi informasi penting pada dokumen administratif secara otomatis.

4.4.3 Analisis Data dan Evaluasi Hasil Pengujian

1. Analisis Hasil Pengujian Fungsional

Berdasarkan hasil pengujian fungsional yang telah dilakukan pada seluruh fitur utama sistem, diperoleh hasil bahwa seluruh skenario pengujian berhasil dijalankan sesuai dengan keluaran yang diharapkan. Fitur autentikasi pengguna, unggah dokumen, ekstraksi teks menggunakan OCR, preprocessing teks, identifikasi entitas menggunakan model Named Entity Recognition (NER), pembentukan ringkasan dokumen, serta penyajian hasil melalui REST API dapat berjalan tanpa ditemukan kegagalan yang menyebabkan proses analisis berhenti. Hasil tersebut menunjukkan bahwa kebutuhan fungsional yang telah dirumuskan pada tahap perancangan sistem berhasil diimplementasikan dengan baik.

Beberapa temuan penting diperoleh selama proses pengujian. Pertama, modul OCR yang menggunakan PaddleOCR mampu mengekstraksi teks dari dokumen administratif dengan baik sehingga informasi pada dokumen dapat diteruskan ke tahap pemrosesan berikutnya. Kedua, proses preprocessing berhasil melakukan

normalisasi teks dan mengurangi karakter yang tidak diperlukan sehingga meningkatkan kualitas masukan bagi model NER. Ketiga, integrasi antara model NER dan modul summarization berjalan secara otomatis melalui REST API sehingga pengguna hanya perlu mengunggah dokumen dan menekan tombol Generate Summary untuk memperoleh hasil analisis.

Selain itu, implementasi REST API berbasis FastAPI yang di-deploy pada AWS EC2 mampu menangani proses inferensi secara stabil. Hasil analisis dapat dikembalikan dalam format JSON yang berisi informasi hasil ekstraksi entitas serta ringkasan dokumen. Berdasarkan keseluruhan hasil pengujian fungsional, dapat disimpulkan bahwa sistem telah memenuhi kebutuhan operasional yang ditetapkan pada penelitian ini.

2. Analisis Validasi Deteksi *Placeholder*

Evaluasi model dilakukan untuk mengukur kemampuan model IndoBERT hasil fine-tuning dalam mengenali entitas penting pada dokumen administratif. Berdasarkan hasil pengujian menggunakan data uji, model memperoleh nilai Precision sebesar 0,8103, Recall sebesar 0,8868, F1-Score sebesar 0,8468, dan Loss sebesar 0,0705. Nilai tersebut menunjukkan bahwa model memiliki kemampuan yang baik dalam mengidentifikasi informasi penting pada dokumen administratif yang menjadi objek penelitian.

Nilai Recall yang lebih tinggi dibandingkan Precision menunjukkan bahwa model mampu menemukan sebagian besar entitas yang terdapat pada dokumen. Hal ini mengindikasikan bahwa kemungkinan entitas penting terlewat relatif kecil. Namun demikian, masih terdapat sejumlah prediksi yang kurang tepat sehingga menyebabkan nilai Precision sedikit lebih rendah. Kondisi tersebut menunjukkan bahwa model cenderung lebih agresif dalam mendeteksi entitas sehingga menghasilkan beberapa kesalahan klasifikasi pada kategori tertentu.

Analisis yang lebih rinci dilakukan menggunakan confusion matrix untuk mengetahui distribusi prediksi pada setiap label. Berdasarkan hasil confusion matrix, seluruh label awal entitas atau kategori B-Label berhasil dikenali dengan

tingkat akurasi yang sangat tinggi bahkan mencapai 100% pada sebagian besar kategori. Hasil ini menunjukkan bahwa model mampu menentukan awal suatu entitas secara konsisten pada berbagai jenis dokumen administratif.

Meskipun demikian, beberapa kategori masih menunjukkan performa yang relatif lebih rendah dibandingkan kategori lainnya. Label I-PERIHAL memperoleh tingkat akurasi sebesar 78,0%, sedangkan label I-LOKASI memperoleh tingkat akurasi sebesar 40,0%. Kesalahan pada kategori tersebut sebagian besar disebabkan oleh variasi format penulisan yang tinggi pada dokumen administratif. Pada kategori lokasi, misalnya, penulisan dapat berupa nama ruangan, gedung, alamat lengkap, ataupun kombinasi beberapa lokasi sekaligus. Variasi tersebut menyebabkan model mengalami kesulitan dalam menentukan batas entitas secara konsisten.

Temuan ini sejalan dengan hasil pengujian pada dokumen TOR, Lampiran, dan Transkrip yang masih menunjukkan beberapa kasus missing label. Namun demikian, secara keseluruhan model tetap mampu menghasilkan identifikasi entitas yang baik dan mendukung proses pembentukan ringkasan dokumen secara otomatis. Dengan nilai F1-Score sebesar 84,68%, model yang dikembangkan dapat dikategorikan memiliki performa yang baik untuk digunakan pada tugas identifikasi teks variabel dinamis pada dokumen administratif.

3. Analisis *User Acceptance Testing*

Berdasarkan hasil User Acceptance Testing yang telah dilakukan, seluruh skenario pengujian memperoleh status Sesuai atau Sesuai dengan Catatan, tanpa adanya skenario yang memperoleh status Tidak Sesuai. Hasil ini menunjukkan bahwa sistem yang dikembangkan telah memenuhi kebutuhan pengguna dan mampu digunakan untuk membantu proses analisis dokumen administratif secara otomatis.

Beberapa temuan penting diperoleh selama pelaksanaan UAT. Pertama, pengguna menilai bahwa proses analisis dokumen yang dilakukan melalui satu tombol Generate Summary memberikan kemudahan dalam memperoleh informasi penting dari dokumen administratif tanpa harus membaca keseluruhan isi dokumen secara manual. Kedua, pengguna menyatakan bahwa hasil ringkasan

yang dihasilkan sistem telah mampu merepresentasikan informasi utama yang terdapat pada dokumen sumber. Ketiga, keluaran dalam format JSON dinilai memudahkan integrasi sistem dengan aplikasi lain pada masa mendatang.

Pengujian terhadap berbagai jenis dokumen administratif seperti Nota Dinas, Surat Keterangan Lulus, Surat Keterangan, Surat Permohonan, Surat Undangan, Pengumuman, dan Pengalihan Perkuliahan menunjukkan hasil yang memuaskan. Sistem mampu menghasilkan ringkasan yang sesuai dengan isi dokumen serta menampilkan entitas penting yang berhasil diekstraksi oleh model NER.

Meskipun demikian, pengguna masih menemukan beberapa kasus missing label pada dokumen TOR, Lampiran, dan Transkrip. Kondisi ini menyebabkan ketiga skenario tersebut memperoleh status Sesuai dengan Catatan. Berdasarkan hasil observasi, permasalahan tersebut berkaitan dengan variasi struktur dokumen yang berbeda dari mayoritas data pelatihan sehingga model mengalami kesulitan dalam mengenali beberapa entitas secara konsisten.

Selain memberikan penilaian terhadap sistem, pengguna juga memberikan beberapa masukan untuk pengembangan lebih lanjut. Masukan tersebut meliputi penambahan variasi data pelatihan untuk meningkatkan kemampuan generalisasi model, optimalisasi proses OCR pada dokumen hasil pemindaian berkualitas rendah, serta perluasan dukungan terhadap jenis dokumen administratif lainnya. Secara keseluruhan, hasil UAT menunjukkan bahwa sistem telah diterima dengan baik oleh pengguna dan dinilai mampu membantu proses identifikasi informasi penting pada dokumen administratif secara otomatis.

4. Evaluasi Pencapaian Tujuan Penelitian

Berdasarkan keseluruhan hasil pengujian (pengujian fungsional *black box*, validasi akurasi algoritma deteksi kolom isian, dan *User Acceptance Testing*) seluruh lima tujuan penelitian yang telah ditetapkan pada sub-bab 1.4 dapat dievaluasi ketercapaiannya sebagaimana dirangkum pada Tabel 17

Tabel 4.11 Tabel Evaluasi Pencapainan Tujuan Penenlitian

Tujuan	Deskripsi	Bukti Ketercapaian	Status

Tujuan	Deskripsi	Bukti Ketercapaian	Status
Tujuan 1	Membangun sistem analisis dokumen administratif berbasis OCR, NER, dan summarization	Sistem berhasil mengunggah dan memproses dokumen PDF maupun DOCX, melakukan ekstraksi teks, identifikasi entitas, serta menghasilkan ringkasan dokumen melalui REST API	Tercapai
Tujuan 2	Mengembangkan model Named Entity Recognition (NER) untuk identifikasi teks variabel dinamis pada dokumen administratif	Model IndoBERT hasil fine-tuning berhasil mengenali entitas NOMOR_SURAT, JENIS_DOKUMEN, TANGGAL, PENGIRIM, PENERIMA, PERIHAL, LOKASI, WAKTU, dan ISI dengan F1-Score terbaik sebesar 88,% pada Run ID #13	Tercapai
Tujuan 3	Mengimplementasikan proses ekstraksi teks dari berbagai format dokumen administratif	Sistem berhasil melakukan ekstraksi teks dari PDF digital menggunakan PyMuPDF, DOCX menggunakan python-docx, dan PDF hasil pemindaian menggunakan PaddleOCR	Tercapai
Tujuan 4	Menghasilkan ringkasan dokumen administratif secara otomatis	Sistem mampu menghasilkan ringkasan dokumen dalam format JSON berdasarkan hasil	Tercapai

Tujuan	Deskripsi	Bukti Ketercapaian	Status
	berdasarkan hasil analisis dokumen	identifikasi entitas dan deteksi informasi penting pada dokumen	
Tujuan 5	evaluasi sistem menggunakan data nyata serta pengguna akhir	Evaluasi model menggunakan Precision, Recall, F1-Score, dan Confusion Matrix menunjukkan performa yang baik, serta seluruh skenario User Acceptance Testing (UAT) memperoleh status Sesuai atau Sesuai dengan Catatan	Tercapai

Seluruh lima tujuan penelitian dinyatakan tercapai. Keterbatasan yang ditemukan selama pengujian yaitu respons yang sesekali lambat pada fitur kolaboratif dan ringkasan AI akibat kondisi jaringan di lokasi pengujian, ketidakmampuan memproses scanned PDF, dan belum adanya pembatasan domain email bersifat dapat diatasi melalui penyesuaian infrastruktur dan pengembangan lanjutan, dan tidak menghambat fungsi inti sistem yang menjadi tujuan penelitian ini.

BAB V
PENUTUP

5.1 Kesimpulan

Penelitian ini berhasil mengembangkan sistem analisis dokumen administratif berbasis OCR (PaddleOCR), NER (IndoBERT fine-tuning), dan ringkasan otomatis. Model NER mencapai performa terbaik dengan F1-score 95,15% (Run ID #13). Sistem mampu mengekstrak teks dari PDF digital, PDF scan, dan DOCX, mengidentifikasi entitas administratif (nomor surat, tanggal, pengirim, penerima, perihal, lokasi), serta menghasilkan ringkasan dokumen dalam format JSON. Hasil pengujian fungsional dan User Acceptance Testing (UAT) menunjukkan bahwa sistem berfungsi sesuai kebutuhan dan diterima baik oleh pengguna, meskipun masih terdapat keterbatasan pada dokumen dengan struktur kompleks (TOR, lampiran, transkrip). Seluruh tujuan penelitian tercapai.

5.2 Saran

Untuk pengembangan lebih lanjut, disarankan agar jumlah dan variasi dataset pelatihan diperbanyak, terutama untuk entitas LOKASI dan PERIHAL yang masih memiliki tingkat akurasi lebih rendah, sehingga model dapat belajar lebih banyak pola penulisan dokumen administratif dari berbagai instansi. Penelitian selanjutnya juga dapat mengoptimalkan pra-pemrosesan citra pada dokumen hasil pemindaian berkualitas rendah, misalnya dengan teknik denoising dan deskewing, guna meningkatkan kualitas ekstraksi teks oleh PaddleOCR. Selain itu, metode ringkasan dapat ditingkatkan dari ekstraktif menjadi abstraktif berbasis transformer agar ringkasan yang dihasilkan lebih alami dan informatif. Terakhir, sistem ini berpotensi dikembangkan menjadi platform yang lebih komprehensif dengan menambahkan fitur klasifikasi dokumen, pencarian semantik, serta integrasi dengan sistem administrasi institusi, kemudian diuji coba pada lingkungan operasional nyata dengan jumlah pengguna yang lebih besar.

DAFTAR PUSTAKA

Ramdhani, T.W. (2025) Pendekatan Hybrid Rule Based dan Deep Learning untuk Named Entity Recognition pada Dokumen Kepegawaian Pemerintah di Indonesia, Disertasi, Fakultas Ilmu Komputer Universitas Indonesia. Tersedia di: https://www.lib.ui.ac.id.

Tandiono, S.M. (2025) Evaluasi Optimalisasi Metode Ekstraksi Teks Berbasiskan Teknologi OCR NER dan LLM, MBKM Report, Universitas Multimedia Nusantara. Tersedia di: https://kc.umn.ac.id/id/eprint/36386/ (Diakses: 9 Juni 2026).

Muchtar, N.U., Arifianto, D. dan Umlasari, R. (2025) 'Analisis Kinerja Transformer Untuk Named Entity Recognition (NER) Menggunakan IndoBERT Pada Transkrip Video Politik Berbahasa Indonesia', Discovery : Jurnal Ilmu Pengetahuan, 10(2), pp. 180-199. doi: 10.33752/jd.v10i2.10133.

Hasanah, U. dkk. (2025) PENERAPAN OPTICAL CHARACTER RECOGNITION (OCR) DAN NATURAL LANGUAGE PROCESSING (NLP) UNTUK PENGELOLAAN INFORMASI DOKUMEN (STUDI KASUS PADA KAPEL ST YOHANES RASUL PAROKI ST YUSUP KARANGPILANG), Buku karya ilmiah, Universitas Telkom. Tersedia di: https://openlibrary.telkomuniversity.ac.id .

Cui, C. dkk. (2025) PaddleOCR-VL: Boosting Multilingual Document Parsing via a 0.9B Ultra-Compact Vision-Language Model, arXiv preprint, Tersedia di: https://arxiv.org/abs/2510.14528 (Diakses: 9 Juni 2026).

Hidayat, T., Nugroho, A. dan Wibowo, B. (2025) 'Extractive Indonesian Automated Text Summarization with IndoBERT and One-Dimensional Convolutional Neural Network', dalam Lecture Notes in Computer Science, Springer. doi: 10.1007/978-3-031-89045-5_8.

Liu, Z., Shi, K. dan Chen, N.F. (2025) 'Transformer-based document-level discourse processing: Exploiting prior language knowledge and hierarchical parsing', Computer Speech & Language, 94, p. 101809.

A. Yani and A. S. Alimsyah, "Pengembangan Teknologi Internet dalam Administrasi Persuratan: Membangun Sistem Surat Otomatis," Jurnal Citra Pendidikan, vol. 5, no. 4, pp. 99-109, 2025.

Zain, M.A., Nugroho, A. dan Lestari, D. (2025) 'Data Augmentation using Large Language Models for Indonesian Named Entity Recognition', Journal of ICT Research and Applications, 19(1), pp. 45-58.

Al Faraby, S., & Romadhony, A. (2022). Pengaruh Distribusi Panjang Data Teks pada Klasifikasi: Sebuah Studi Awal. Jurnal Media Informatika Budidarma, 6(3), 1501. https://doi.org/10.30865/mib.v6i3.4259

Binmakhashen, G. M., & Mahmoud, S. A. (2020). Document layout analysis: a comprehensive survey. ACM Computing Surveys (CSUR), 52(6), 1-36.

DAFTAR RIWAYAT HIDUP

Lahir di Manado, 13-12-2004

Penulis merupakan mahasiswa Program Studi Teknik Informatika, Jurusan Teknik Informatika dan Komputer, Politeknik Negeri Jakarta. Penulis menempuh pendidikan dasar di SDS ANGKASA 1 kemudian melanjutkan pendidikan di SMPN 50 JAKARTA dan SMAS ANGKASA 1. Pada tahun 2022, penulis melanjutkan pendidikan Diploma IV/Sarjana Terapan pada Program Studi Teknik Informatika, Politeknik Negeri Jakarta.

LAMPIRAN

Lampiran 1 Transkrip Wawancara

Judul Penelitian	PENGEMBANGAN SISTEM CERDAS UNTUK MANAJEMEN, ANALISIS, DAN OTOMATISASI DOKUMEN BERBASIS WEB
Nama Peneliti	Aurio Hendrianoko Rajaa Muhammad Nathan
Hari/Tanggal	Selasa/Mar 03 2026
Narasumber	Mba Faras
Daftar Pertanyaan	Bagaimana alur SOP pada penulisan surat? Apakah sudah menggunakan sistem otomatisasi seperti mail merge atau masih manual? Bagaimana cara penyimpanan dan pengelolaan dokumen saat ini? Apa saja jenis-jenis dokumen yang sering dibuat? Bagaimana perbedaan format antar jenis dokumen tersebut? Bagaimana proses pembuatan dokumen khusus seperti surat peringatan (SP)? Berapa jumlah dokumen yang diproses dalam satu periode (misalnya per tahun)? Bagaimana proses pembuatan dokumen magang? Bagaimana cara distribusi dokumen kepada pihak terkait? Apakah masih memerlukan hard copy atau sudah full digital? Seberapa sering template dokumen diperbarui? Apa kendala utama dalam proses pengelolaan dokumen saat ini?

Hasil Transkrip	Peneliti: Mba, saya mau tanya terkait alur kerja pembuatan dokumen di sini. Secara umum, workflow-nya itu seperti apa dari awal sampai dokumen jadi? Narasumber: Kalau secara umum sih, biasanya kita mulai dari kebutuhan dulu ya. Misalnya ada permintaan surat, nanti kita lihat jenisnya apa, terus kita pakai template yang sudah ada. Kadang kalau banyak datanya, kita pakai mail merge buat generate banyak dokumen sekaligus. Peneliti: Berarti sudah ada automation sedikit ya, seperti mail merge? Narasumber: Iya, tapi belum semuanya. Masih banyak juga yang manual, apalagi kalau datanya beda-beda atau inputnya tidak terstruktur. Peneliti: Kalau penyimpanan dokumennya sendiri gimana? Narasumber: Disimpan di folder, jadi ada central repository gitu. Tapi ya masih folder biasa, belum ada sistem khusus. Peneliti: Untuk kolaborasi atau kerja bareng tim, apakah sudah ada sistemnya? Narasumber: Belum terlalu, biasanya masih sharing file aja. Peneliti: Apa saja jenis-jenis dokumen yang sering dibuat di sini? Narasumber: Banyak sih. Yang sering itu seperti surat tugas, surat keterangan, surat keputusan (SK), sama notulensi. Peneliti: Kalau untuk yang eksternal? Narasumber: Ada juga undangan, surat pengadaan, perbaikan kelas, sama surat jawaban. Kadang juga surat pengantar, misalnya untuk MoU atau

	keperluan lain. Peneliti: Untuk formatnya apakah berbeda-beda? Narasumber: Iya, tapi sebagian besar sudah ada templatenya. Cuma tetap perlu disesuaikan lagi. Peneliti: Kalau untuk dokumen seperti surat peringatan (SP), bagaimana formatnya? Narasumber: SP itu biasanya ada SP1, SP2, SP3, formatnya hampir sama, cuma beda di tingkatannya. Peneliti: Dalam satu tahun kira-kira berapa banyak dokumen yang dibuat? Narasumber: Kurang lebih sekitar 1500 dokumen per tahun. Peneliti: Untuk kasus tertentu seperti magang, itu prosesnya bagaimana? Narasumber: Kalau magang, biasanya mahasiswa isi Google Form dulu. Setelah itu, datanya kita ambil, terus kita buatkan suratnya secara manual. Peneliti: Berarti belum otomatis ya dari form ke dokumen? Narasumber: Belum, masih input satu-satu. Peneliti: Untuk distribusi dokumen, biasanya dikirim bagaimana? Narasumber: Sekarang kebanyakan lewat email. Jadi sudah tidak perlu hard copy untuk beberapa jenis dokumen. Peneliti: Kalau untuk template sendiri, apakah sering berubah?

	Narasumber: Iya, biasanya ada update. Misalnya sekarang pakai template 2025–2026, nanti bisa berubah lagi.

Bukti Dokumentasi	

Lampiran 2 Dokumentasi Pelaksanaan *User Acceptance Testing*