

NER untuk OCR Document

Ekstraksi Entitas Cerdas dengan Ollama, Python, dan Ubuntu
26.04

Onno W. Purbo
onno@indo.net.id
onno@itts.ac.id
Twitter onnowpurbo

Institut Teknologi Tangerang Selatan (ITTS)
Dinas KOMINFO Tangerang Selatan
OnnoCenter
2026

Lisensi: Karya ini dilisensikan di bawah **Creative Commons Attribution (CC BY) 4.0 International**. Anda bebas menggunakan, membagikan, dan mengadaptasi karya ini untuk tujuan apa pun, termasuk komersial, dengan tetap mencantumkan atribusi yang layak kepada penulis.

Disclaimer: Buku ini disusun untuk tujuan edukasi, dokumentasi, dan refleksi publik berdasarkan sumber-sumber yang dianggap kredibel pada saat penulisan. Penulis telah berupaya menjaga akurasi, namun sangat mungkin masih terdapat kekurangan atau kekeliruan; karena itu, buku ini tidak dimaksudkan sebagai satu-satunya rujukan yang final, dan penulis terbuka terhadap koreksi serta masukan yang membangun

Dokumen yang hanya disimpan adalah arsip, tetapi dokumen yang dipahami, dihubungkan, dan dimanfaatkan melalui AI akan menjadi pengetahuan yang menggerakkan keputusan, inovasi, dan masa depan.

Kata Pengantar

Dokumen adalah aset yang paling banyak dimiliki organisasi, tetapi sering kali menjadi aset yang paling sedikit dimanfaatkan. Ribuan halaman KTP, surat, kontrak, invoice, arsip sejarah, hingga rekam medis tersimpan dalam bentuk PDF atau hasil pemindaian yang sulit dicari, dianalisis, dan diintegrasikan ke dalam proses bisnis. Di sinilah **OCR (Optical Character Recognition)**, **NER (Named Entity Recognition)**, dan **LLM (Large Language Model)** membuka peluang baru untuk mengubah dokumen menjadi pengetahuan yang hidup dan bernilai strategis.

Buku ini disusun dengan pendekatan yang praktis, modern, dan berorientasi implementasi. Seluruh pembahasan dibangun menggunakan ekosistem **open source**, mulai dari Ubuntu, Python, PaddleOCR, Ollama, Pydantic, hingga MariaDB, sehingga pembaca dapat membangun solusi secara mandiri tanpa ketergantungan pada layanan eksternal. Fokus utama buku ini bukan sekadar menjalankan model AI, melainkan membangun **pipeline ekstraksi informasi yang dapat diaudit, diukur, diamankan, dan dioperasikan pada lingkungan produksi**.

Lebih dari sekadar buku teknis, buku ini merupakan ajakan untuk membangun **kedaulatan teknologi dan kedaulatan data**. Organisasi yang mampu memahami, mengelola, dan mengolah dokumen nya sendiri akan memiliki keunggulan kompetitif yang jauh lebih kuat dibanding organisasi yang hanya menjadi pengguna teknologi. Di tengah perkembangan pesat AI, kemampuan membangun solusi lokal yang transparan, efisien, dan berkelanjutan akan menjadi aset strategis yang sangat berharga.

Semoga buku ini dapat menjadi panduan sekaligus inspirasi bagi administrator jaringan, praktisi keamanan siber, pengembang perangkat lunak, peneliti, mahasiswa, dan siapa saja yang ingin membangun AI yang benar-benar bermanfaat. **Teknologi terbaik bukanlah teknologi yang paling rumit, melainkan teknologi yang mampu memberdayakan manusia, memperkuat organisasi, dan membuka akses pengetahuan bagi lebih banyak orang.**

Jakarta, Juni 2026
Onno W. Purbo

Executive Summary

Di era transformasi digital, tantangan terbesar organisasi bukan lagi kekurangan data, melainkan ketidakmampuan mengubah dokumen menjadi pengetahuan yang dapat dimanfaatkan secara cepat, akurat, dan strategis. Ribuan halaman hasil pemindaian, PDF, arsip administrasi, kontrak, invoice, rekam medis, hingga dokumen sejarah sering tersimpan sebagai informasi pasif yang sulit dicari, dianalisis, dan diintegrasikan ke dalam proses bisnis maupun pengambilan keputusan. Buku ini memperkenalkan pendekatan modern untuk mengatasi masalah tersebut melalui integrasi **OCR (Optical Character Recognition)**, **NER (Named Entity Recognition)**, dan **LLM (Large Language Model)** yang berjalan sepenuhnya pada infrastruktur lokal menggunakan ekosistem *open source*. Pembaca diajak membangun solusi end-to-end mulai dari ekstraksi teks menggunakan PaddleOCR, pemrosesan entitas cerdas menggunakan Ollama dan model bahasa modern, validasi data menggunakan Pydantic, hingga penyimpanan terstruktur ke MariaDB. Tidak berhenti pada tingkat laboratorium, buku ini juga membahas arsitektur produksi yang mencakup *batch processing*, *queue management*, *vector search*, *structured output*, keamanan data, monitoring performa, optimasi GPU, serta praktik terbaik deployment dan operasional. Fokus utama buku ini adalah membangun **AI yang dapat diaudit, dikendalikan, dan dimiliki sendiri**, sehingga organisasi tidak hanya menjadi pengguna teknologi, tetapi mampu menguasai dan mengembangkannya sesuai kebutuhan. Melalui berbagai proyek nyata pada dokumen administrasi, keuangan, arsip sejarah, dan domain khusus lainnya, pembaca akan memahami bagaimana mengubah dokumen menjadi aset strategis yang menghasilkan wawasan, efisiensi operasional, dan nilai bisnis yang terukur. Pada akhirnya, buku ini bukan sekadar panduan teknis OCR dan NER, melainkan sebuah peta jalan untuk membangun **kedaulatan data, kemandirian teknologi, dan fondasi AI lokal yang transparan, aman, hemat biaya, serta mampu memberdayakan organisasi dan masyarakat secara berkelanjutan**.

Daftar Isi

Kata Pengantar.....	4
Executive Summary.....	5
Daftar Isi.....	6
BAB 1 - Fondasi Konsep NER, OCR, dan LLM.....	17
1.1 Pendahuluan.....	17
1.2 Apa Itu Named Entity Recognition (NER).....	18
1.3 Evolusi Teknologi NER.....	19
Era Rule-Based.....	19
Era Machine Learning.....	19
Era Deep Learning.....	20
Era LLM-Powered NER.....	20
Zero-Shot.....	20
Few-Shot.....	20
Structured Output.....	21
1.4 Mengapa LLM Lebih Unggul untuk NER.....	21
Fleksibel.....	22
Multilingual.....	22
Context-Aware.....	22
Cepat Diadaptasi.....	22
1.5 OCR dan Tantangan Dunia Nyata.....	23
Scan Buruk.....	23
Noise.....	23
Layout Kompleks.....	23
Dokumen Indonesia.....	23
1.6 Mengapa OCR Saja Tidak Cukup.....	24
1.7 OCR + NER Sebagai Game Changer.....	25
Administrasi Pemerintah.....	25
Keuangan.....	25
Kesehatan.....	25
Pendidikan.....	25
Sejarah.....	25
1.8 Arsitektur Open Source Lokal.....	26
1.9 Structured Output dan JSON.....	27
1.10 Entity Linking dan Validasi.....	28
1.11 Kedaulatan Data dan AI Lokal.....	28
1.12 Penutup.....	29
BAB 2 - Menyiapkan Lingkungan Ubuntu 26.04 LTS.....	30
2.1 Mengapa Ubuntu 26.04 LTS?.....	30
2.2 Arsitektur Sistem yang Akan Dibangun.....	31
2.3 Persiapan Perangkat Keras.....	32
Minimum.....	32
Direkomendasikan.....	32

Untuk LLM Besar.....	32
2.4 Instalasi Ubuntu 26.04.....	32
2.5 Update Sistem.....	33
2.6 Monitoring Resource Linux.....	33
2.7 Konfigurasi Swap.....	34
2.8 Optimasi HugePages.....	34
2.9 Memastikan cgroup v2 Aktif.....	35
2.10 Instalasi Driver GPU.....	35
NVIDIA.....	35
AMD.....	35
2.11 Instalasi Ollama.....	36
2.12 Struktur Penyimpanan Model.....	36
2.13 Manajemen Model Ollama.....	36
2.14 Menjalankan Ollama Sebagai Service.....	37
2.15 Konfigurasi Multi-GPU.....	37
2.16 Instalasi Python.....	37
2.17 Virtual Environment.....	38
2.18 Instalasi Library Python.....	38
2.19 Instalasi MariaDB.....	38
2.20 Hardening MariaDB.....	39
2.21 Membuat Database NER.....	39
2.22 Pengujian Koneksi Python ke MariaDB.....	39
2.23 Checklist Akhir Bab.....	40
Sistem.....	40
AI.....	40
Python.....	40
Database.....	40
2.24 Penutup.....	40
BAB 3 - OCR Handal – Dari Gambar dan PDF ke Teks Bersih.....	41
3.1 Mengapa OCR Menjadi Fondasi NER.....	41
3.2 Evolusi OCR Modern.....	41
Generasi Awal.....	41
Generasi Machine Learning.....	42
Generasi Deep Learning.....	42
3.3 Memilih OCR yang Tepat.....	43
PaddleOCR.....	43
Tesseract OCR.....	43
EasyOCR.....	44
DocTR.....	44
3.4 Perbandingan Tool OCR.....	45
3.5 Struktur Pipeline OCR.....	45
3.6 Instalasi Library OCR.....	46
3.7 Membaca Gambar Pertama.....	46
3.8 Masalah Umum Dokumen.....	46

3.9 Preprocessing Menggunakan OpenCV.....	47
3.10 Grayscale Conversion.....	47
3.11 Denoising.....	47
3.12 Thresholding dan Binarization.....	48
3.13 Adaptive Threshold.....	48
3.14 Contrast Enhancement.....	48
3.15 Deskewing.....	49
3.16 OCR untuk PDF.....	49
PDF Digital.....	49
PDF Scan.....	49
3.17 Menangani Dokumen Multi Halaman.....	50
3.18 Multi-Column Detection.....	50
3.19 Table Detection.....	50
3.20 OCR untuk Dokumen Indonesia.....	51
3.21 Text Cleaning.....	51
3.22 Pipeline OCR End-to-End.....	52
3.23 Evaluasi OCR.....	52
3.24 Character Error Rate (CER).....	53
3.25 Word Error Rate (WER).....	53
3.26 Mengapa CER dan WER Penting.....	53
3.27 Strategi Produksi.....	54
3.28 Checklist Akhir Bab.....	54
3.29 Penutup.....	54
BAB 4 - Ollama untuk NER – Dari Prompt ke Structured Output.....	55
4.1 Dari OCR ke NER.....	55
4.2 Mengapa Ollama?.....	56
Berjalan Lokal.....	56
Open Source Friendly.....	56
Mendukung Banyak Model.....	56
Mudah Diotomasi.....	56
4.3 Arsitektur Ollama.....	57
4.4 Cara Kerja NER Berbasis LLM.....	58
4.5 Structured Output.....	59
4.6 Mengapa JSON Sangat Penting.....	59
4.7 Pydantic Sebagai Lapisan Validasi.....	60
4.8 Model yang Direkomendasikan Tahun 2026.....	60
4.9 Llama 3.x.....	60
4.10 Qwen3.....	61
4.11 Gemma 4.....	61
4.12 Phi-4.....	61
4.13 UniversalNER.....	62
4.14 Perbandingan Model.....	62
4.15 Prompt Engineering untuk NER.....	63
4.16 Teknik Zero-Shot.....	63

4.17 Teknik Few-Shot.....	63
4.18 Chain-of-Thought.....	64
4.19 Prompt Produksi.....	64
4.20 Menjalankan Ollama dari Python.....	65
4.21 Structured Output dengan Pydantic.....	65
4.22 Menangani Hallucination.....	65
4.23 Validasi Pasca Ekstraksi.....	66
4.24 Benchmark Sederhana.....	66
4.25 Mengukur Latency.....	66
4.26 Benchmark yang Perlu Dicatat.....	67
4.27 Strategi Produksi yang Direkomendasikan.....	67
4.28 Checklist Akhir Bab.....	68
4.29 Penutup.....	68
BAB 5 - Membangun Core Pipeline Python + Ollama.....	69
5.1 Dari Komponen Menjadi Sistem.....	69
5.2 Filosofi Pipeline Produksi.....	70
5.3 Struktur Folder Proyek.....	71
5.4 Library yang Digunakan.....	72
5.5 Membangun Schema dengan Pydantic.....	73
5.6 Mengapa Schema Penting.....	74
5.7 Membuat Prompt Produksi.....	75
5.8 Integrasi Ollama dengan Python.....	76
5.9 Fungsi Reusable NER.....	76
5.10 Implementasi extract_entities_ner().....	77
5.11 Generic Schema Approach.....	77
5.12 Structured Output dengan Pydantic.....	78
5.13 Entity Standar.....	78
5.14 Entity Custom.....	79
5.15 Parsing JSON Output.....	79
5.16 Error Handling.....	80
5.17 Retry Logic.....	80
5.18 Timeout Protection.....	81
5.19 Confidence Scoring.....	81
5.20 Validasi NIK.....	81
5.21 Logging Produksi.....	82
5.22 Contoh Log Produksi.....	82
5.23 Koneksi MariaDB.....	82
5.24 Membuat Tabel Entity.....	83
5.25 Generate Query INSERT Otomatis.....	83
5.26 Pipeline Lengkap.....	84
5.27 Pipeline Produksi yang Direkomendasikan.....	85
5.28 Checklist Akhir Bab.....	86
5.29 Penutup.....	86
BAB 6 - Proyek Hands-on 1 – Dokumen Administrasi Indonesia.....	87

6.1 Dari Konsep Menjadi Solusi Nyata.....	87
6.2 Studi Kasus Dokumen Indonesia.....	87
KTP.....	88
Surat Resmi.....	88
Ijazah.....	88
Sertifikat.....	88
6.3 Struktur Folder Proyek.....	89
6.4 Alur Kerja Sistem.....	90
6.5 Membuat Schema KTP.....	90
6.6 Membuat Schema Surat.....	91
6.7 Membuat Schema Ijazah.....	91
6.8 OCR dengan PaddleOCR.....	91
6.9 Menguji OCR.....	92
6.10 Prompt NER KTP.....	92
6.11 Menjalankan Ollama.....	93
6.12 Pipeline OCR + NER.....	93
6.13 Custom Entity Indonesia.....	94
6.14 Normalisasi Data.....	94
6.15 Normalisasi Tanggal.....	95
6.16 Normalisasi Nomor.....	95
6.17 Deduplikasi Data.....	95
6.18 Validasi NIK.....	96
6.19 Validasi Nomor Surat.....	96
6.20 Membuat Database.....	96
6.21 Tabel KTP.....	96
6.22 Tabel Surat.....	97
6.23 Tabel Ijazah.....	97
6.24 Insert Otomatis.....	98
6.25 Batch Processing.....	98
6.26 Logging Produksi.....	99
6.27 Script CLI Lengkap.....	99
6.28 Arsitektur Produksi.....	100
6.29 Checklist Akhir Bab.....	100
6.30 Penutup.....	101
BAB 7 - Proyek Hands-on 2 – Dokumen Bisnis dan Keuangan.....	102
7.1 Mengapa Dokumen Bisnis Lebih Sulit?.....	102
7.2 Jenis Dokumen yang Akan Diproses.....	103
Invoice.....	103
Kwitansi.....	103
Purchase Order.....	103
Kontrak.....	103
7.3 Arsitektur Sistem.....	104
7.4 Struktur Folder Proyek.....	105
7.5 Tantangan OCR pada Invoice.....	106

7.6 Schema Invoice.....	106
7.7 Schema Item Invoice.....	107
7.8 Relasi Header dan Detail.....	107
7.9 Table Extraction.....	108
7.10 OCR dan Layout Analysis.....	108
7.11 Prompt Invoice.....	109
7.12 Prompt Kontrak.....	109
7.13 Entity Relationship Extraction.....	110
7.14 Relasi dalam Kontrak.....	110
7.15 Normalisasi Nominal.....	111
7.16 Validasi Nominal.....	111
7.17 Validasi Tanggal.....	111
7.18 Validasi Perhitungan Invoice.....	112
7.19 Validasi Item Invoice.....	112
7.20 Database Header Invoice.....	113
7.21 Database Detail Invoice.....	113
7.22 Konsep Transaction Safety.....	113
7.23 Contoh Transaction.....	114
7.24 Mengapa Transaction Penting.....	114
7.25 Bulk Processing.....	115
7.26 Batch Insert.....	115
7.27 Logging Audit.....	115
7.28 Dashboard Statistik Sederhana.....	116
7.29 Arsitektur Produksi.....	116
7.30 Checklist Akhir Bab.....	117
7.31 Penutup.....	117
BAB 8 - Proyek Hands-on 3 – Arsip Sejarah, Medis, dan Skala Besar.....	118
8.1 Ketika OCR Masuk Dunia Nyata.....	118
8.2 Karakteristik Arsip Sejarah.....	119
Noise Tinggi.....	119
Kertas Rusak.....	119
Font Kuno.....	119
Layout Tidak Konsisten.....	119
8.3 Karakteristik Dokumen Medis.....	120
8.4 Mengapa Domain-Specific NER Penting.....	121
8.5 Arsitektur Produksi Skala Besar.....	121
8.6 Struktur Folder Produksi.....	122
8.7 Domain-Specific Schema.....	122
8.8 Schema Arsip Sejarah.....	123
8.9 Prompt Domain Khusus.....	123
8.10 Pipeline Pemrosesan Massal.....	124
8.11 Konsep Queue.....	124
8.12 Worker OCR.....	125
8.13 Worker NER.....	125

8.14 Paralel Processing.....	125
8.15 GPU Scheduling.....	126
8.16 Monitoring GPU.....	126
8.17 Batch Processing.....	126
8.18 Mengukur Throughput.....	127
8.19 Caching Hasil OCR.....	127
8.20 Caching Hasil NER.....	127
8.21 Menggunakan Hash Dokumen.....	128
8.22 Fine-Tuning Ringan dengan LoRA.....	128
8.23 Kapan LoRA Dibutuhkan?.....	128
8.24 Dataset Fine-Tuning.....	129
8.25 Risiko Fine-Tuning.....	129
Dataset Terlalu Sedikit.....	129
Data Kotor.....	129
Overfitting.....	129
8.26 Vector Search untuk Arsip.....	129
8.27 Arsitektur Vector Search.....	130
8.28 Embedding Lokal.....	131
8.29 Integrasi NER dan Vector Search.....	131
8.30 Analitik Skala Besar.....	132
Arsip Sejarah.....	132
Medis.....	132
8.31 Disaster Recovery.....	132
8.32 Audit dan Kepatuhan.....	133
8.33 Arsitektur Produksi Lengkap.....	133
8.34 Checklist Akhir Bab.....	134
8.35 Penutup.....	134
BAB 9 - Optimasi, Keamanan, dan Produksi.....	135
9.1 Dari Laboratorium ke Produksi.....	135
9.2 Filosofi Produksi.....	136
9.3 Parameter yang Harus Dimonitor.....	136
Performa.....	136
Infrastruktur.....	136
Akurasi.....	136
Operasional.....	136
9.4 Monitoring CPU.....	137
9.5 Monitoring Memori.....	137
9.6 Monitoring GPU.....	137
9.7 Monitoring Storage.....	138
9.8 Mengukur Latency.....	138
9.9 Mengukur Throughput.....	138
9.10 Mengukur Akurasi OCR.....	139
9.11 Mengukur Akurasi NER.....	139
9.12 Mengapa Data Lokal Penting.....	140

9.13 Enkripsi Penyimpanan.....	140
Enkripsi Disk.....	140
Enkripsi Backup.....	140
Enkripsi Database.....	140
9.14 Hardening Server.....	141
9.15 Isolasi Layanan.....	141
9.16 Logging Produksi.....	142
9.17 Audit Trail.....	142
9.18 Deployment dengan Container.....	143
9.19 Contoh Dockerfile.....	143
9.20 Menjalankan Sebagai Service.....	144
9.21 Wrapper API dengan FastAPI.....	144
9.22 Arsitektur API Produksi.....	145
9.23 Antarmuka Web.....	145
9.24 Masalah OOM (Out of Memory).....	146
9.25 Hallucination.....	146
9.26 JSON Tidak Valid.....	147
9.27 OCR Buruk.....	147
9.28 Masalah Multibahasa.....	147
9.29 Masalah Koneksi MariaDB.....	148
9.30 Masalah Kinerja Menurun.....	148
9.31 Backup Produksi.....	148
9.32 High Availability Sederhana.....	149
9.33 Checklist Produksi.....	149
Infrastruktur.....	149
AI.....	149
Keamanan.....	149
Operasional.....	149
9.34 Arsitektur Produksi Lengkap.....	150
9.35 Penutup.....	150
BAB 10 - Masa Depan NER Lokal dan Inspirasi.....	151
10.1 Kita Baru Memulai.....	151
10.2 Evolusi OCR dan NER.....	152
10.3 Tren 2026–2027.....	153
10.4 Multimodal Model.....	153
10.5 Dari OCR ke Document Understanding.....	154
10.6 Small Language Models (SLM).....	154
10.7 Mengapa SLM Penting untuk Indonesia.....	155
10.8 Agentic Workflow.....	155
10.9 Dari Ekstraksi Menjadi Otomasi.....	156
10.10 Knowledge Graph.....	156
10.11 AI untuk Arsip Nasional.....	157
10.12 AI untuk Rumah Sakit.....	157
10.13 AI untuk Pendidikan.....	157

10.14 AI untuk Pemerintah Daerah.....	158
10.15 Mengapa Open Source Penting.....	158
10.16 Komunitas Open Source Indonesia.....	159
10.17 Komunitas Global.....	159
10.18 Ide Proyek Lanjutan.....	159
10.19 Tantangan yang Akan Muncul.....	160
10.20 Berpikir Seperti Engineer.....	160
10.21 Dokumentasikan Semua.....	160
10.22 Bagikan Pipeline Anda.....	161
10.23 Bangun Dataset Lokal.....	161
10.24 Membangun Kedaulatan AI.....	161
10.25 Peran Administrator Jaringan dan Praktisi Siber.....	162
10.26 Checklist Setelah Buku Selesai.....	162
10.27 Pesan Penutup.....	163
Selamat Berkarya.....	163
Lampiran A - Daftar Perintah CLI Lengkap Ubuntu + Ollama.....	164
A.1 Update Sistem.....	164
A.2 Monitoring Server.....	164
A.3 GPU.....	164
A.4 Python Environment.....	164
A.5 Library Python Utama.....	165
A.6 Instalasi Ollama.....	165
A.7 Manajemen Model Ollama.....	165
A.8 MariaDB.....	166
Lampiran B - Template Prompt NER Terbaik + Pydantic Schema.....	167
B.1 Prompt NER Umum.....	167
B.2 Prompt KTP.....	168
B.3 Prompt Invoice.....	168
B.4 Pydantic Schema Umum.....	169
B.5 Pydantic Schema KTP.....	169
B.6 Pydantic Schema Invoice.....	170
B.7 Structured Output Ollama.....	171
Lampiran C - MariaDB Schema & Contoh Query.....	172
C.1 Membuat Database.....	172
C.2 Membuat User.....	172
C.3 Tabel Dokumen.....	172
C.4 Tabel Entity Umum.....	172
C.5 Tabel KTP.....	173
C.6 Tabel Invoice.....	173
C.7 Tabel Item Invoice.....	174
C.8 Insert Dokumen.....	174
C.9 Insert Entity.....	174
C.10 Insert Aman dengan Duplikasi.....	174
Lampiran D - Daftar Model Ollama Rekomendasi 2026.....	176

D.1 Prinsip Memilih Model.....	176
D.2 Model Rekomendasi.....	176
D.3 Rekomendasi Praktis.....	177
Laptop / CPU Only.....	177
Server GPU 8–12 GB VRAM.....	177
Server GPU Lebih Besar.....	177
Ekstraksi JSON Ketat.....	177
Lampiran E - Resources, Dataset Open Source, dan FAQ Troubleshooting.....	178
E.1 Resources Open Source.....	178
OCR.....	178
LLM Lokal.....	178
Python.....	178
Database.....	178
E.2 Dataset Open Source.....	179
E.3 FAQ Troubleshooting.....	179
1. Ollama tidak jalan.....	179
2. Model lambat.....	179
3. Output JSON rusak.....	179
4. Model mengarang data.....	180
5. OCR buruk.....	180
6. MariaDB gagal konek.....	180
7. Database duplikat.....	180
8. Proses ribuan dokumen lambat.....	180
9. Memory habis.....	181
10. Hasil NER tidak konsisten.....	181
Penutup Lampiran.....	181
Glossary.....	182
Referensi.....	187
Tentang Penulis.....	190

BAB 1 - Fondasi Konsep NER, OCR, dan LLM

1.1 Pendahuluan

Setiap organisasi menghasilkan dokumen dalam jumlah sangat besar setiap hari. Surat resmi, kontrak, invoice, laporan keuangan, identitas penduduk, rekam medis, tiket, formulir, hingga arsip sejarah terus bertambah.

Masalahnya bukan pada penyimpanan dokumen.

Masalah sesungguhnya adalah bagaimana **mengubah dokumen menjadi pengetahuan yang dapat diproses mesin**.

Sebagian besar informasi penting masih terkunci di dalam:

- PDF hasil scan
- Foto dokumen
- Arsip lama
- Berkas hasil cetak
- Formulir kertas
- Dokumen tidak terstruktur

Manusia dapat membaca dokumen tersebut dengan mudah.

Komputer tidak.

Di sinilah teknologi **Optical Character Recognition (OCR)** dan **Named Entity Recognition (NER)** menjadi sangat penting.

OCR mengubah gambar menjadi teks.

NER mengubah teks menjadi informasi terstruktur.

Ketika keduanya dikombinasikan dengan **Large Language Model (LLM)** modern, dokumen yang sebelumnya hanya menjadi arsip pasif dapat berubah menjadi sumber data strategis yang siap digunakan oleh aplikasi, basis data, mesin analitik, dan sistem kecerdasan buatan.

1.2 Apa Itu Named Entity Recognition (NER)

Named Entity Recognition (NER) adalah proses mengidentifikasi dan mengklasifikasikan entitas penting di dalam teks.

Contoh:

```
Nama saya Budi Santoso.  
Saya bekerja di Institut Teknologi Tangerang Selatan.  
Alamat kantor berada di Tangerang Selatan.
```

Hasil NER:

```
{  
  "PERSON": "Budi Santoso",  
  "ORGANIZATION": "Institut Teknologi Tangerang Selatan",  
  "LOCATION": "Tangerang Selatan"  
}
```

NER berusaha menjawab pertanyaan:

- Siapa?
- Di mana?
- Kapan?
- Organisasi apa?
- Nomor apa?
- Produk apa?
- Berapa nilainya?

Dengan kata lain, NER mengubah teks menjadi data yang dapat diproses lebih lanjut.

1.3 Evolusi Teknologi NER

Era Rule-Based

Generasi pertama NER menggunakan aturan manual.

Contoh:

```
if "PT." in text:  
    entity = "ORGANIZATION"
```

Kelebihan:

- Mudah dipahami
- Cepat
- Tidak membutuhkan dataset besar

Kekurangan:

- Sulit dipelihara
- Tidak fleksibel
- Banyak kasus gagal

Pendekatan ini cocok untuk lingkungan yang sangat terbatas tetapi tidak cocok untuk skala nasional.

Era Machine Learning

Pendekatan berikutnya menggunakan algoritma statistik.

Contoh:

- HMM
- CRF
- SVM

Model belajar dari data pelatihan.

Kelebihan:

- Lebih akurat
- Tidak terlalu bergantung aturan manual

Kekurangan:

- Membutuhkan dataset berlabel
- Sulit membuat entity baru

Era Deep Learning

Kemunculan:

- RNN
- LSTM
- BiLSTM
- Transformer

meningkatkan akurasi secara signifikan.

Model mulai memahami konteks kalimat.

Contoh:

"Kuningan"

dapat dikenali sebagai:

- Kabupaten Kuningan
- Jalan Kuningan
- Nama gedung

tergantung konteks.

Era LLM-Powered NER

Model modern mampu melakukan NER tanpa pelatihan ulang.

Teknik yang umum digunakan:

Zero-Shot

Model belum pernah melihat contoh sebelumnya.

Contoh:

Ekstrak nomor kontrak dari dokumen berikut.

Model langsung memahami tugas.

Few-Shot

Model diberi beberapa contoh.

Contoh:

Nomor kontrak: ABC-123

Output:

```
{  
  "contract_number": "ABC-123"  
}
```

Akurasi biasanya meningkat signifikan.

Structured Output

Model tidak menghasilkan teks bebas.

Model dipaksa menghasilkan:

```
{  
  "nama": "",  
  "alamat": "",  
  "nomor_ktp": ""  
}
```

Pendekatan ini menjadi fondasi utama sistem produksi modern.

1.4 Mengapa LLM Lebih Unggul untuk NER

Model NER tradisional biasanya hanya mengenali entitas yang sudah dilatih sebelumnya.

Misalnya:

- PERSON
- LOCATION
- ORGANIZATION

Namun dunia nyata jauh lebih kompleks.

Sebuah organisasi mungkin membutuhkan:

- Nomor kontrak
- Nomor invoice
- Nomor pelanggan
- Nomor rekening
- Nomor KTP
- Nomor NIP
- Nomor sertifikat

LLM mampu mengenali entitas baru hanya melalui instruksi.

Contoh:

Ekstrak:

- nomor pelanggan
- nomor meter listrik
- daya listrik

tanpa perlu pelatihan ulang.

Keunggulan utama:

Fleksibel

Entity baru dapat dibuat kapan saja.

Multilingual

Mampu memahami:

- Indonesia
- Inggris
- Arab
- Jepang
- Campuran bahasa

Context-Aware

Mampu memahami makna berdasarkan konteks.

Cepat Diadaptasi

Tidak perlu membuat dataset ribuan baris terlebih dahulu.

1.5 OCR dan Tantangan Dunia Nyata

OCR adalah teknologi yang mengubah gambar menjadi teks.

Secara teori sederhana.

Dalam praktik sangat sulit.

Dokumen nyata biasanya memiliki masalah:

Scan Buruk

- Buram
- Miring
- Resolusi rendah

Noise

- Noda
- Stempel
- Tanda tangan

Layout Kompleks

- Multi kolom
- Tabel
- Header
- Footer

Dokumen Indonesia

- Campuran bahasa
- Singkatan lokal
- Format administrasi unik

Handwriting

Tulisan tangan masih menjadi tantangan besar hingga saat ini.

1.6 Mengapa OCR Saja Tidak Cukup

Misalnya OCR menghasilkan:

Nama : Budi Santoso

NIK : 317301xxxxxxxxxx

Alamat : Tangerang Selatan

OCR hanya menghasilkan teks.

OCR tidak memahami:

- mana nama
- mana alamat
- mana NIK

Di sinilah NER bekerja.

NER memahami struktur informasi.

Hasil akhirnya:

```
{  
  "nama": "Budi Santoso",  
  "nik": "317301xxxxxxxxxx",  
  "alamat": "Tangerang Selatan"  
}
```

Informasi tersebut dapat langsung masuk ke database.

1.7 OCR + NER Sebagai Game Changer

Kombinasi OCR dan NER membuka peluang besar.

Administrasi Pemerintah

- Digitalisasi arsip
- Pengelolaan surat
- Ekstraksi metadata

Keuangan

- Invoice
- Faktur
- Kwitansi

Kesehatan

- Rekam medis
- Surat rujukan
- Hasil laboratorium

Pendidikan

- Ijazah
- Transkrip
- Sertifikat

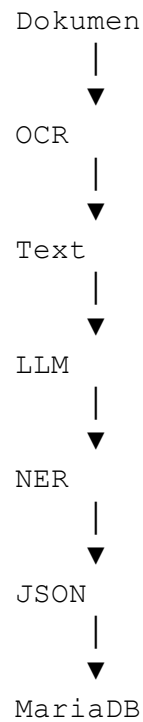
Sejarah

- Arsip kolonial
- Manuskrip lama
- Dokumen budaya

Dokumen yang sebelumnya memerlukan ribuan jam kerja manual dapat diproses secara otomatis.

1.8 Arsitektur Open Source Lokal

Pendekatan buku ini menggunakan arsitektur sederhana namun kuat.



Komponen utama:

Ubuntu 26.04

Platform server yang stabil.

Python

Bahasa utama orkestrasi.

OCR Engine

Untuk ekstraksi teks.

Ollama

Mesin inferensi model lokal.

MariaDB

Penyimpanan hasil ekstraksi.

Seluruh proses dapat berjalan tanpa ketergantungan layanan eksternal.

1.9 Structured Output dan JSON

Kesalahan terbesar dalam implementasi AI produksi adalah menggunakan output bebas.

Contoh buruk:

Nama pelanggan adalah Budi.
Alamatnya di Tangerang Selatan.

Sulit diproses aplikasi.

Sebaliknya:

```
{  
  "nama": "Budi",  
  "alamat": "Tangerang Selatan"  
}
```

lebih mudah digunakan.

Keuntungan:

- Konsisten
- Mudah divalidasi
- Mudah disimpan
- Mudah dicari
- Mudah diintegrasikan

JSON menjadi bahasa universal antar sistem.

1.10 Entity Linking dan Validasi

Ekstraksi saja tidak cukup.

Data harus diverifikasi.

Contoh:

```
{
  "kota": "Tangsel"
}
```

dapat dinormalisasi menjadi:

```
{
  "kota": "Tangerang Selatan"
}
```

Teknik ini disebut entity linking.

Tujuannya:

- Mengurangi duplikasi
- Meningkatkan kualitas data
- Mempermudah analitik

Prinsip yang sama digunakan dalam sistem pencarian modern dan knowledge graph.

1.11 Kedaulatan Data dan AI Lokal

Banyak organisasi mulai menyadari bahwa data adalah aset strategis.

Mengirim dokumen sensitif ke layanan eksternal menimbulkan risiko:

- Kebocoran data
- Ketergantungan vendor
- Biaya berulang
- Kepatuhan regulasi

Pendekatan AI lokal memberikan keuntungan:

Data Tetap di Dalam Organisasi

Tidak keluar ke pihak lain.

Biaya Lebih Rendah

Tidak ada biaya per token.

Dapat Dikustomisasi

Model dapat disesuaikan dengan kebutuhan organisasi.

Mendukung Kemandirian Teknologi

Organisasi memiliki kendali penuh terhadap sistem.

1.12 Penutup

OCR mengubah gambar menjadi teks.

NER mengubah teks menjadi informasi.

LLM mengubah informasi menjadi pengetahuan yang dapat dipahami mesin.

Ketika ketiganya digabungkan dalam arsitektur *open source* berbasis Ubuntu, Python, Ollama, dan MariaDB, organisasi memperoleh kemampuan untuk membangun sistem ekstraksi dokumen yang **murah, mandiri, aman, dan siap produksi**.

Bab ini memberikan fondasi konseptual yang diperlukan sebelum masuk ke tahap implementasi. Pada bab berikutnya, pembaca akan menyiapkan lingkungan Ubuntu 26.04, Python, basis data, dan mesin inferensi lokal sebagai fondasi pembangunan pipeline OCR + NER modern.

BAB 2 - Menyiapkan Lingkungan Ubuntu 26.04 LTS

2.1 Mengapa Ubuntu 26.04 LTS?

Banyak proyek AI gagal bukan karena model yang buruk.

Masalah terbesar justru ada pada:

- Sistem operasi tidak stabil
- Dependensi berantakan
- Driver GPU bermasalah
- Lingkungan kerja tidak konsisten

Karena itu kita memerlukan fondasi yang kuat.

Ubuntu 26.04 LTS dipilih karena:

- Stabil untuk server dan workstation
- Siklus dukungan jangka panjang
- Dokumentasi sangat lengkap
- Dukungan komunitas luas
- Kompatibel dengan mayoritas perangkat AI modern

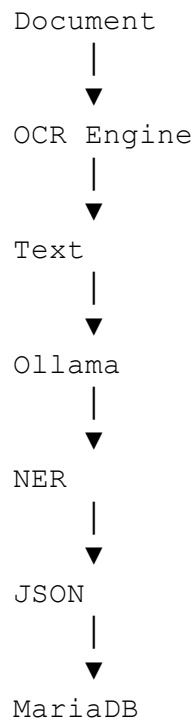
Bagi administrator jaringan dan praktisi keamanan siber, kestabilan lebih penting dibanding fitur eksperimental.

Prinsip utama:

Model AI dapat diganti kapan saja. Infrastruktur yang stabil jauh lebih berharga.

2.2 Arsitektur Sistem yang Akan Dibangun

Target akhir buku ini adalah membangun pipeline:



Komponen utama:

Komponen	Fungsi
Ubuntu 26.04	Sistem operasi
Python	Orkestrasi
Ollama	Inferensi LLM
OCR Engine	Ekstraksi teks
MariaDB	Penyimpanan
OpenCV	Preprocessing gambar

Seluruh proses berjalan secara lokal.

Tidak ada ketergantungan layanan eksternal.

2.3 Persiapan Perangkat Keras

Minimum

Untuk laboratorium kecil:

- CPU 8 Core
- RAM 16 GB
- SSD 512 GB

Direkomendasikan

Untuk produksi ringan:

- CPU 12–16 Core
- RAM 32 GB
- SSD NVMe 1 TB

Untuk LLM Besar

- RAM 64 GB+
- GPU VRAM 16 GB+

Semakin besar model yang digunakan, semakin besar kebutuhan memori.

2.4 Instalasi Ubuntu 26.04

Lakukan instalasi bersih.

Partisi yang direkomendasikan:

Mount Point	Ukuran
/boot	2 GB
/	100 GB
/var	100 GB
/home	Sisa disk
swap	16–32 GB

Untuk server AI:

Pisahkan /var.

Karena model Ollama dan log biasanya tumbuh sangat besar.

2.5 Update Sistem

Langkah pertama setelah instalasi:

```
sudo apt update
sudo apt upgrade -y
sudo apt autoremove -y
```

Verifikasi versi:

```
lsb_release -a
```

Cek kernel:

```
uname -r
```

Cek informasi sistem:

```
hostnamectl
```

2.6 Monitoring Resource Linux

Administrator wajib memahami kondisi sistem.

CPU

```
lscpu
```

Memory

```
free -h
```

Storage

```
df -h
```

I/O

```
iostat -xz 1
```

Proses

```
htop
```

Tool yang direkomendasikan:

```
sudo apt install htop sysstat ntop
```

2.7 Konfigurasi Swap

LLM sangat rakus memori.

Cek swap:

```
swapon --show
```

Membuat swap tambahan:

```
sudo fallocate -l 32G /swapfile
```

Atur permission:

```
sudo chmod 600 /swapfile
```

Aktifkan:

```
sudo mkswap /swapfile  
sudo swapon /swapfile
```

Tambahkan ke:

```
/etc/fstab
```

Agar aktif saat boot.

2.8 Optimasi HugePages

Model besar mendapat manfaat dari HugePages.

Cek:

```
cat /proc/meminfo | grep Huge
```

Aktifkan:

```
sudo sysctl -w vm.nr_hugepages=2048
```

Permanen:

```
sudo nano /etc/sysctl.conf
```

Tambahkan:

```
vm.nr_hugepages=2048
```

2.9 Memastikan cgroup v2 Aktif

Sebagian besar aplikasi modern memanfaatkan cgroup v2.

Verifikasi:

```
stat -fc %T /sys/fs/cgroup
```

Output:

```
cgroup2fs
```

Jika muncul nilai tersebut maka sistem siap.

2.10 Instalasi Driver GPU

NVIDIA

Cek perangkat:

```
lspci | grep VGA
```

Deteksi driver:

```
ubuntu-drivers devices
```

Instal:

```
sudo ubuntu-drivers autoinstall
```

Verifikasi:

```
nvidia-smi
```

AMD

Instal:

```
sudo apt install mesa-utils
```

Cek:

```
glxinfo | grep OpenGL
```

Monitoring:

```
radeontop
```

2.11 Instalasi Ollama

Ollama menjadi fondasi utama NER berbasis LLM.

Instalasi:

```
curl -fsSL https://ollama.com/install.sh | sh
```

Verifikasi:

```
ollama version
```

Cek service:

```
systemctl status ollama
```

Aktifkan saat boot:

```
sudo systemctl enable ollama
```

2.12 Struktur Penyimpanan Model

Direkomendasikan:

```
/opt/models/  
/opt/models/ollama/  
/opt/logs/  
/opt/projects/
```

Buat direktori:

```
sudo mkdir -p /opt/models  
sudo mkdir -p /opt/projects
```

Atur permission:

```
sudo chown -R $USER:$USER /opt/projects
```

2.13 Manajemen Model Ollama

Menampilkan model:

```
ollama list
```

Menarik model:

```
ollama pull llama3
```

Menjalankan model:

```
ollama run llama3
```

Menghapus model:

```
ollama rm llama3
```

Informasi model:

```
ollama show llama3
```

2.14 Menjalankan Ollama Sebagai Service

Verifikasi:

```
systemctl status ollama
```

Restart:

```
sudo systemctl restart ollama
```

Melihat log:

```
journalctl -u ollama -f
```

Administrator Linux harus terbiasa membaca log.
Sebagian besar masalah dapat ditemukan dari log.

2.15 Konfigurasi Multi-GPU

Cek GPU:

```
nvidia-smi
```

Menampilkan GPU secara kontinu:

```
watch -n 1 nvidia-smi
```

Monitoring lebih nyaman:

```
nvidia-smi
```

Untuk server besar, pastikan distribusi beban GPU dipantau secara rutin.

2.16 Instalasi Python

Verifikasi:

```
python3 --version
```

Instal paket penting:

```
sudo apt install \
python3-pip \
python3-venv \
python3-dev \
build-essential
```

Upgrade pip:

```
pip install --upgrade pip
```

2.17 Virtual Environment

Buat lingkungan kerja:

```
mkdir ner-project  
cd ner-project
```

Buat virtual environment:

```
python3 -m venv .venv
```

Aktifkan:

```
source .venv/bin/activate
```

Verifikasi:

```
which python
```

2.18 Instalasi Library Python

Library inti:

```
pip install ollama  
pip install pydantic  
pip install pymysql  
pip install opencv-python  
pip install pandas  
pip install numpy
```

Instal sekaligus:

```
pip install \  
ollama \  
pydantic \  
pymysql \  
opencv-python \  
pandas \  
numpy
```

2.19 Instalasi MariaDB

Instal:

```
sudo apt install mariadb-server
```

Aktifkan:

```
sudo systemctl enable mariadb
```

Start:

```
sudo systemctl start mariadb
```

Verifikasi:

```
systemctl status mariadb
```

2.20 Hardening MariaDB

Jalankan:

```
sudo mysql_secure_installation
```

Lakukan:

- Set password root
- Hapus anonymous user
- Nonaktifkan remote root
- Hapus test database

Langkah sederhana ini meningkatkan keamanan secara signifikan.

2.21 Membuat Database NER

Masuk:

```
sudo mysql
```

Buat database:

```
CREATE DATABASE nerdb;
```

Buat user:

```
CREATE USER 'neruser'@'localhost'  
IDENTIFIED BY 'passwordku';
```

Berikan hak akses:

```
GRANT ALL PRIVILEGES  
ON nerdb.*  
TO 'neruser'@'localhost';
```

Aktifkan:

```
FLUSH PRIVILEGES;
```

2.22 Pengujian Koneksi Python ke MariaDB

Contoh sederhana:

```
import pymysql  
  
conn = pymysql.connect(  
    host="localhost",  
    user="neruser",  
    password="passwordku",  
    database="nerdb"  
)  
  
print("Koneksi berhasil")  
conn.close()
```

Jika berhasil, fondasi database siap digunakan.

2.23 Checklist Akhir Bab

Pastikan seluruh komponen berikut sudah berjalan:

Sistem

- Ubuntu 26.04
- Kernel terbaru
- Swap aktif
- HugePages aktif

AI

- Ollama berjalan
- Model berhasil diunduh
- GPU terdeteksi

Python

- Virtual environment aktif
- Semua library terinstal

Database

- MariaDB aktif
- User database tersedia
- Python dapat terhubung

2.24 Penutup

Bab ini membangun fondasi teknis yang akan digunakan sepanjang buku. Administrator jaringan dan praktisi keamanan siber harus memahami bahwa **keberhasilan sistem OCR + NER tidak dimulai dari model AI, tetapi dari infrastruktur yang stabil, dapat direproduksi, aman, dan mudah dipelihara.**

Pada bab berikutnya, kita mulai membangun komponen pertama pipeline, yaitu **OCR modern berbasis open source**, untuk mengubah gambar, PDF, formulir, dan dokumen hasil scan menjadi teks berkualitas tinggi yang siap diproses oleh LLM dan NER.

BAB 3 - OCR Handal – Dari Gambar dan PDF ke Teks Bersih

3.1 Mengapa OCR Menjadi Fondasi NER

Banyak proyek NER gagal bukan karena model NER yang buruk.

Penyebab utama biasanya adalah:

- OCR salah membaca karakter
- Tata letak dokumen rusak
- Tabel tidak dikenali
- Dokumen miring
- Hasil OCR penuh noise

Prinsip sederhana:

Garbage In, Garbage Out

Jika OCR menghasilkan teks yang buruk, maka NER dan LLM akan menghasilkan informasi yang buruk pula.

Karena itu OCR harus menjadi perhatian utama sebelum berbicara mengenai AI.

3.2 Evolusi OCR Modern

OCR telah berkembang sangat jauh.

Generasi Awal

Pendekatan berbasis:

- Template matching
- Rule-based pattern

Akurasi rendah.

Tidak cocok untuk dokumen dunia nyata.

Generasi Machine Learning

Mulai menggunakan:

- Feature extraction
- Statistical recognition

Lebih baik tetapi masih terbatas.

Generasi Deep Learning

Menggunakan:

- CNN
- CRNN
- Transformer
- Vision Language Model

Mampu menangani:

- Layout kompleks
- Multi bahasa
- Tabel
- Dokumen rusak

Mayoritas OCR modern tahun 2026 menggunakan pendekatan ini.

3.3 Memilih OCR yang Tepat

Tidak ada OCR terbaik untuk semua kasus.
Setiap mesin memiliki kelebihan dan kekurangan.

PaddleOCR

Pilihan utama untuk buku ini.

Keunggulan:

- Akurasi tinggi
- Mendukung banyak bahasa
- Deteksi layout sangat baik
- Table recognition
- PDF processing
- Komunitas besar

Sangat cocok untuk:

- Invoice
- KTP
- Surat resmi
- Arsip organisasi

Tesseract OCR

Veteran dunia OCR.

Keunggulan:

- Ringan
- Stabil
- Mudah dipasang
- Banyak bahasa

Kekurangan:

- Lemah pada layout kompleks
- Sulit menangani tabel modern

Cocok untuk:

- Dokumen sederhana
- Server kecil

EasyOCR

Keunggulan:

- Instalasi mudah
- Dukungan banyak bahasa
- Baik untuk prototipe

Kekurangan:

- Akurasi di bawah PaddleOCR

DocTR

Fokus pada:

- Dokumen kompleks
- PDF modern
- Layout understanding

Keunggulan:

- Arsitektur modern
- Dukungan transformer

Cocok untuk riset dan eksperimen.

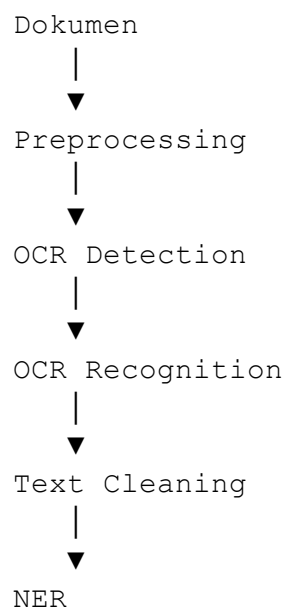
3.4 Perbandingan Tool OCR

Fitur	PaddleOCR	Tesseract	EasyOCR	DocTR
Bahasa Indonesia	Sangat Baik	Baik	Baik	Sangat Baik
Tabel	Sangat Baik	Terbatas	Terbatas	Baik
Layout Kompleks	Sangat Baik	Rendah	Sedang	Sangat Baik
Kecepatan	Tinggi	Tinggi	Sedang	Sedang
GPU	Ya	Tidak	Ya	Ya
Produksi	Sangat Cocok	Cocok	Prototipe	Riset

Untuk mayoritas proyek OCR + NER, PaddleOCR merupakan pilihan paling seimbang.

3.5 Struktur Pipeline OCR

Pipeline yang akan digunakan:



Setiap tahap memiliki pengaruh besar terhadap akurasi akhir.

3.6 Instalasi Library OCR

Buat lingkungan kerja:

```
mkdir ocr-lab
cd ocr-lab
```

Aktifkan:

```
source .venv/bin/activate
```

Instal library:

```
pip install paddleocr
pip install paddlepaddle
pip install opencv-python
pip install pdf2image
pip install pillow
pip install numpy
```

3.7 Membaca Gambar Pertama

Contoh sederhana:

```
from paddleocr import PaddleOCR

ocr = PaddleOCR(lang='en')

result = ocr.ocr("contoh.jpg")
print(result)
```

Output berisi:

- Koordinat teks
- Isi teks
- Confidence score

3.8 Masalah Umum Dokumen

Dokumen nyata sering memiliki:

Rotasi

Dokumen miring saat dipindai.

Blur

Kamera tidak fokus.

Noise

Noda atau bintik.

Kontras Rendah

Tulisan sulit dibedakan.

Bayangan

Hasil foto ponsel.

OCR tidak menyukai kondisi ini.

3.9 Preprocessing Menggunakan OpenCV

Preprocessing sering meningkatkan akurasi secara signifikan.

Tujuannya:

- Membersihkan gambar
- Menonjolkan teks
- Mengurangi noise

3.10 Grayscale Conversion

Mengubah gambar menjadi abu-abu.

```
import cv2

img = cv2.imread("scan.jpg")

gray = cv2.cvtColor(
    img,
    cv2.COLOR_BGR2GRAY
)
```

Keuntungan:

- Lebih cepat
- Mengurangi informasi tidak penting

3.11 Denoising

Menghilangkan gangguan visual.

```
denoise = cv2.fastNlMeansDenoising(
    gray
)
```

Cocok untuk:

- Dokumen lama
- Scan berkualitas rendah

3.12 Thresholding dan Binarization

Mengubah gambar menjadi hitam putih.

```
_ , binary = cv2.threshold(  
    gray,  
    150,  
    255,  
    cv2.THRESH_BINARY  
)
```

Manfaat:

- Memperjelas karakter
- Memudahkan OCR

3.13 Adaptive Threshold

Lebih cocok untuk dokumen tidak merata.

```
binary = cv2.adaptiveThreshold(  
    gray,  
    255,  
    cv2.ADAPTIVE_THRESH_GAUSSIAN_C,  
    cv2.THRESH_BINARY,  
    11,  
    2  
)
```

3.14 Contrast Enhancement

Menggunakan CLAHE.

```
clahe = cv2.createCLAHE()  
enhanced = clahe.apply(gray)
```

Hasil:

- Teks lebih tajam
- Background lebih bersih

3.15 Deskewing

Masalah paling sering:

Dokumen miring.

Contoh:

5 derajat

10 derajat

15 derajat

sudah cukup membuat OCR turun drastis.

Deskewing digunakan untuk mengembalikan posisi dokumen.

Keuntungan:

- OCR lebih akurat
- Layout lebih stabil

3.16 OCR untuk PDF

Sebagian besar dokumen organisasi berbentuk PDF.

Ada dua jenis:

PDF Digital

Sudah memiliki teks.

Tidak perlu OCR.

PDF Scan

Berisi gambar.

Harus di-OCR.

Konversi:

```
from pdf2image import convert_from_path

pages = convert_from_path(
    "dokumen.pdf"
)
```

Setiap halaman menjadi gambar.

3.17 Menangani Dokumen Multi Halaman

Contoh:

```
for page in pages:  
    # OCR setiap halaman
```

Strategi:

- Simpan halaman terpisah
- Beri nomor halaman
- Gabungkan hasil akhir

3.18 Multi-Column Detection

Masalah umum:

Kolom kiri

Kolom kanan

OCR sederhana sering membaca secara salah.

Layout detection menjadi penting.

PaddleOCR dan DocTR memiliki kemampuan ini.

3.19 Table Detection

Invoice dan laporan keuangan biasanya berisi tabel.

Contoh:

Produk	Harga
Router	1.000.000
Switch	2.000.000

Jika tabel rusak:

NER akan gagal.

Karena itu deteksi tabel menjadi komponen penting.

3.20 OCR untuk Dokumen Indonesia

Tantangan lokal:

- NIK
- NIP
- NPWP
- Nomor Surat
- Singkatan instansi
- Campuran bahasa

OCR harus mampu mempertahankan format angka dan simbol.

Contoh:

```
3173xxxxxxxxxxxxxx
```

Tidak boleh berubah menjadi:

```
317Bxxxxxxxxxxxxxx
```

Kesalahan satu karakter dapat membuat data tidak valid.

3.21 Text Cleaning

Setelah OCR selesai.

Lakukan normalisasi.

Contoh:

```
text = text.replace("\n", " ")
```

Hilangkan:

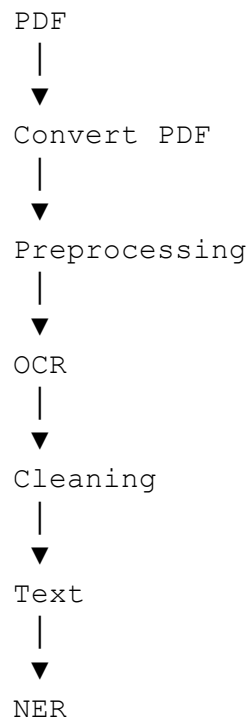
- Karakter aneh
- Spasi ganda
- Simbol tidak relevan

Tujuan:

Membantu NER bekerja lebih baik.

3.22 Pipeline OCR End-to-End

Alur produksi:



Inilah fondasi sistem yang akan dibangun pada bab berikutnya.

3.23 Evaluasi OCR

Banyak organisasi hanya melihat hasil visual.
Pendekatan tersebut tidak cukup.
OCR harus diukur.

3.24 Character Error Rate (CER)

Mengukur kesalahan karakter.

Rumus:

$$CER = (S+D+I) / N$$

Keterangan:

- S = Substitution
- D = Deletion
- I = Insertion
- N = Total karakter

$$CER = (S+D+I) / N$$

Semakin kecil nilainya semakin baik.

Target produksi:

$$CER < 5\%$$

3.25 Word Error Rate (WER)

Mengukur kesalahan kata.

$$WER = (S+D+I) / N$$

Target produksi:

$$WER < 10\%$$

3.26 Mengapa CER dan WER Penting

Contoh:

3173012345678901

OCR menghasilkan:

317301234567B901

Hanya satu karakter salah.

Namun NIK menjadi tidak valid.

Kesalahan kecil dapat menghasilkan kegagalan besar pada NER.

3.27 Strategi Produksi

Untuk sistem produksi:

Tahap 1

OCR

Tahap 2

Validasi OCR

Tahap 3

Text Cleaning

Tahap 4

NER

Tahap 5

Validasi Entity

Tahap 6

Database

Jangan pernah langsung mengirim hasil OCR ke database.

3.28 Checklist Akhir Bab

Pastikan pembaca mampu:

- Menginstal PaddleOCR
- Menginstal OpenCV
- Membaca gambar
- Memproses PDF
- Melakukan deskew
- Melakukan denoise
- Membersihkan teks
- Menghitung CER
- Menghitung WER

Jika seluruh checklist tercapai, maka fondasi OCR sudah siap digunakan.

3.29 Penutup

OCR adalah pintu masuk seluruh proses ekstraksi informasi. Semakin baik kualitas OCR, semakin tinggi kualitas NER dan LLM yang dibangun di atasnya. Praktisi jaringan, keamanan siber, dan administrator Linux perlu memahami bahwa **OCR bukan sekadar membaca teks, tetapi proses rekayasa data yang menentukan keberhasilan seluruh pipeline AI dokumen.**

Pada bab berikutnya, kita akan mulai menggunakan **Ollama dan LLM modern** untuk mengubah teks hasil OCR menjadi **entitas terstruktur berbentuk JSON** yang siap diproses, dicari, dianalisis, dan disimpan ke MariaDB.

BAB 4 - Ollama untuk NER – Dari Prompt ke Structured Output

4.1 Dari OCR ke NER

Pada bab sebelumnya kita berhasil mengubah:

Gambar
PDF
Scan
Dokumen

menjadi teks.

Namun teks masih belum cukup.

Contoh hasil OCR:

Nama : Budi Santoso
NIK : 3173012345678901
Alamat : Tangerang Selatan0

Komputer masih melihatnya sebagai teks biasa.

Tujuan kita adalah mengubahnya menjadi:

```
{  
  "nama": "Budi Santoso",  
  "nik": "3173012345678901",  
  "alamat": "Tangerang Selatan"  
}
```

Format seperti ini dapat:

- Masuk database
- Dicari
- Diolah
- Divalidasi
- Digunakan aplikasi lain

Di sinilah Ollama dan LLM berperan.

4.2 Mengapa Ollama?

Banyak organisasi ingin memanfaatkan AI tetapi menghadapi tantangan:

- Privasi
- Kepatuhan regulasi
- Biaya operasional
- Ketergantungan vendor

Ollama memberikan solusi sederhana.

Keunggulan:

Berjalan Lokal

Data tidak keluar dari organisasi.

Open Source Friendly

Mudah diintegrasikan dengan ekosistem Linux.

Mendukung Banyak Model

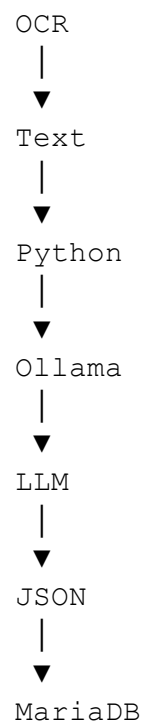
- Llama
- Qwen
- Gemma
- Phi
- Mistral
- DeepSeek
- Model khusus NER

Mudah Diotomasi

Cukup menggunakan REST API atau Python.

4.3 Arsitektur Ollama

Secara sederhana:



Python bertindak sebagai *orchestrator*.

Ollama bertindak sebagai *inference engine*.

Model bertindak sebagai otak sistem.

4.4 Cara Kerja NER Berbasis LLM

NER klasik biasanya membutuhkan:

- Dataset
- Training
- Fine tuning
- Evaluasi

NER berbasis LLM jauh lebih sederhana.

Kita cukup memberikan instruksi.

Contoh:

Ekstrak:

- Nama
- Alamat
- Nomor KTP

Keluarkan dalam JSON.

Model langsung memahami tugas tersebut.

Inilah yang disebut:

Zero-Shot NER

Tidak ada pelatihan tambahan.

4.5 Structured Output

Kesalahan terbesar implementasi LLM adalah membiarkan model menghasilkan teks bebas.

Contoh buruk:

Nama pelanggan adalah Budi Santoso.
Alamat berada di Tangerang Selatan.

Sulit diproses aplikasi.

Pendekatan modern menggunakan:

- JSON Schema
- Pydantic
- Validasi otomatis

Contoh:

```
{  
  "nama": "",  
  "alamat": "",  
  "nik": ""  
}
```

Model dipaksa mengikuti struktur tersebut.

4.6 Mengapa JSON Sangat Penting

JSON menjadi bahasa universal.

Keuntungan:

Konsisten

Output selalu sama.

Mudah Diproses

Python dapat membaca langsung.

Mudah Disimpan

Cocok untuk MariaDB.

Mudah Divalidasi

Kesalahan lebih cepat ditemukan.

Prinsip produksi:

Jangan pernah mengirim output teks bebas ke database.

4.7 Pydantic Sebagai Lapisan Validasi

Pydantic menjadi standar de facto pada aplikasi AI modern.

Contoh schema:

```
from pydantic import BaseModel

class KTP(BaseModel):
    nama: str
    nik: str
    alamat: str
```

Jika output model tidak sesuai schema:

```
{
  "nama": 123
}
```

Pydantic akan menolak.

Ini sangat penting untuk sistem produksi.

4.8 Model yang Direkomendasikan Tahun 2026

Tidak semua model cocok untuk NER.

Kriteria:

- Konsisten menghasilkan JSON
- Cepat
- Multibahasa
- Kuat memahami konteks

4.9 Llama 3.x

Kelebihan:

- Stabil
- Komunitas besar
- Dokumentasi luas

Cocok untuk:

- NER umum
- Dokumen administrasi
- Sistem produksi

Kekurangan:

- Membutuhkan memori relatif besar

4.10 Qwen3

Salah satu model paling menarik untuk NER.

Keunggulan:

- Sangat baik untuk *structured output*
- Kuat pada multilingual
- Cepat

Sangat cocok untuk:

- Bahasa Indonesia
- Dokumen pemerintahan
- Invoice

4.11 Gemma 4

Keunggulan:

- Ringan
- Cepat
- Efisien

Cocok untuk:

- Laptop
- Mini server
- Edge computing

4.12 Phi-4

Kelebihan:

- Ukuran kecil
- Reasoning baik

Cocok untuk:

- Laboratorium
- Eksperimen

4.13 UniversalNER

Model khusus NER.

Keunggulan:

- Fokus ekstraksi entitas
- Akurasi tinggi

Kekurangan:

- Kurang fleksibel dibanding LLM umum

4.14 Perbandingan Model

Model	Akurasi NER	JSON Output	Bahasa Indonesia	Kecepatan
Llama 3.x	Tinggi	Sangat Baik	Baik	Sedang
Qwen3	Sangat Tinggi	Sangat Baik	Sangat Baik	Cepat
Gemma 4	Baik	Baik	Baik	Sangat Cepat
Phi-4	Baik	Baik	Sedang	Cepat
UniversalNER	Sangat Tinggi	Baik	Baik	Cepat

Untuk mayoritas proyek OCR + NER Indonesia, Qwen3 dan Llama 3.x menjadi pilihan utama.

4.15 Prompt Engineering untuk NER

Prompt adalah program baru dalam dunia AI.

Prompt yang buruk menghasilkan output buruk.

Prompt yang baik menghasilkan data berkualitas tinggi.

4.16 Teknik Zero-Shot

Contoh:

Ekstrak:

- nama
- alamat
- nomor ktp

Output JSON saja.

Sederhana.

Cepat.

Cocok untuk eksperimen awal.

4.17 Teknik Few-Shot

Model diberikan contoh.

Input:

Nama: Budi

Output:

```
{  
  "nama": "Budi"  
}
```

Akurasi biasanya meningkat signifikan.

4.18 Chain-of-Thought

Pendekatan ini meminta model berpikir bertahap.

Contoh:

```
Identifikasi seluruh entitas.  
Verifikasi entitas.  
Susun JSON akhir.
```

Berguna untuk:

- Kontrak
- Invoice
- Dokumen hukum

Namun sering menambah waktu inferensi.

4.19 Prompt Produksi

Contoh yang lebih matang:

```
Anda adalah sistem ekstraksi data.
```

```
Ekstrak:
```

```
- nama  
- alamat  
- nomor_ktp
```

```
Keluarkan JSON valid.
```

```
Jangan tambahkan komentar.  
Jangan gunakan markdown.  
Jangan menjelaskan hasil.
```

Prompt sederhana sering lebih baik dibanding prompt yang terlalu panjang.

4.20 Menjalankan Ollama dari Python

Contoh dasar:

```
from ollama import chat

response = chat(
    model="qwen3",
    messages=[
        {
            "role": "user",
            "content": "Halo"
        }
    ]
)

print(response.message.content)
```

Ini adalah fondasi integrasi seluruh sistem.

4.21 Structured Output dengan Pydantic

Contoh schema:

```
from pydantic import BaseModel

class DataKTP(BaseModel):
    nama: str
    nik: str
    alamat: str
```

Keuntungan:

- Validasi otomatis
- Konsistensi
- Siap database

4.22 Menangani Hallucination

Masalah terbesar LLM adalah hallucination.

Contoh:

Dokumen tidak memiliki NIK.
Model justru membuat NIK sendiri.
Ini sangat berbahaya.

Solusi:

- Gunakan schema ketat
- Gunakan validasi
- Gunakan aturan bisnis

4.23 Validasi Pasca Ekstraksi

Contoh:

NIK Indonesia:
16 digit

Jika tidak 16 digit:
REJECT

Tanggal:
YYYY-MM-DD

Jika tidak sesuai:
REJECT

Jangan percaya 100% pada model.

4.24 Benchmark Sederhana

Kita perlu mengukur.

Bukan menebak.

Parameter:

Akurasi

Jumlah entity benar.

Latency

Waktu inferensi.

Throughput

Dokumen per menit.

Konsistensi JSON

Persentase output valid.

4.25 Mengukur Latency

Contoh:

```
import time
start = time.time()

# inferensi
end = time.time()
print(end-start)
```

Target produksi biasanya:

- < 5 detik per dokumen sederhana
- < 15 detik untuk dokumen kompleks

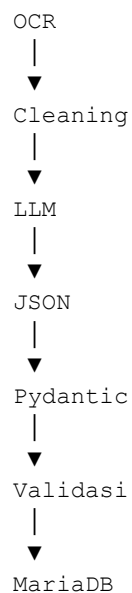
4.26 Benchmark yang Perlu Dicatat

Parameter	Nilai
Model	Qwen3
CPU	Ryzen 7
RAM	32 GB
GPU	RTX 3060
Dokumen	KTP
Latency	x detik
Akurasi	x%

Dokumentasi benchmark sangat penting.

4.27 Strategi Produksi yang Direkomendasikan

Pipeline yang direkomendasikan:



Jangan langsung:

OCR → Database

Karena akan menghasilkan banyak data kotor.

4.28 Checklist Akhir Bab

Pastikan pembaca mampu:

- Menginstal model Ollama
- Memilih model yang tepat
- Membuat prompt NER
- Membuat schema Pydantic
- Menghasilkan JSON konsisten
- Mengukur latency
- Mengukur akurasi
- Mengurangi *hallucination*

Jika seluruh poin tercapai, maka fondasi NER berbasis LLM sudah siap digunakan.

4.29 Penutup

OCR menghasilkan teks. Namun teks belum memiliki makna bisnis. Ollama dan LLM modern memungkinkan kita mengubah teks tersebut menjadi **entitas terstruktur yang dapat langsung diproses aplikasi, basis data, dan sistem analitik**. Kunci keberhasilan bukan hanya memilih model terbaik, tetapi membangun *structured output*, validasi ketat, dan benchmark yang terukur.

Pada bab berikutnya kita akan mulai membangun **core pipeline Python + Ollama**, membuat fungsi reusable untuk ekstraksi entitas, validasi JSON menggunakan Pydantic, serta integrasi langsung ke MariaDB sehingga seluruh proses OCR → NER → Database dapat berjalan otomatis dari ujung ke ujung.

BAB 5 - Membangun Core Pipeline Python + Ollama

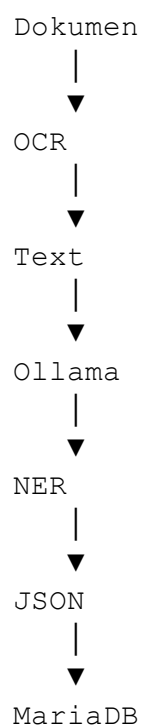
5.1 Dari Komponen Menjadi Sistem

Pada bab sebelumnya kita telah memiliki:

- Ubuntu 26.04
- Python
- Ollama
- OCR Engine
- MariaDB

Namun semuanya masih berdiri sendiri.

Tujuan bab ini adalah membangun *core engine* yang mampu:



Inilah fondasi seluruh proyek pada bab selanjutnya.

5.2 Filosofi Pipeline Produksi

Banyak contoh AI hanya cocok untuk demonstrasi.

Masalah umum:

- Tidak ada validasi
- Tidak ada logging
- Tidak ada retry
- Tidak ada error handling
- Tidak ada integrasi database

Dalam produksi prinsipnya berbeda:

Lebih baik lambat tetapi konsisten daripada cepat tetapi tidak dapat dipercaya.

Karena itu pipeline harus:

- Reusable
- Modular
- Mudah diuji
- Mudah dipelihara
- Mudah diperluas

5.3 Struktur Folder Proyek

Direkomendasikan:

```
ner-project/  
├── data/  
├── logs/  
├── models/  
├── output/  
├── config/  
├── app/  
│   ├── ner.py  
│   ├── database.py  
│   ├── schemas.py  
│   ├── prompts.py  
│   └── utils.py  
├── main.py  
└── requirements.txt
```

Keuntungan:

- Terorganisir
- Mudah dikembangkan
- Mudah dikelola tim

5.4 Library yang Digunakan

Instalasi:

```
pip install ollama
pip install pydantic
pip install pymysql
pip install pandas
pip install loguru
```

Library utama:

Library	Fungsi
ollama	Akses model
pydantic	Validasi data
pymysql	Database
pandas	Manipulasi data
loguru	Logging

5.5 Membangun Schema dengan Pydantic

Langkah pertama adalah mendefinisikan struktur data.

File:

```
app/schemas.py
```

Contoh:

```
from pydantic import BaseModel

class EntityDocument(BaseModel):
    nama: str | None = None
    alamat: str | None = None
    nik: str | None = None
    nip: str | None = None
    tanggal: str | None = None
```

Schema menjadi kontrak antara:

- OCR
- LLM
- Database

5.6 Mengapa Schema Penting

Tanpa schema:

```
{  
  "Nama": "Budi"  
}
```

Kadang menjadi:

```
{  
  "nama_lengkap": "Budi"  
}
```

Kadang:

```
{  
  "person": "Budi"  
}
```

Masalah:

Database tidak konsisten.

Dengan schema:

```
{  
  "nama": "Budi"  
}
```

selalu sama.

5.7 Membuat Prompt Produksi

File:

```
app/prompts.py
```

Contoh:

```
PROMPT_NER = """
Anda adalah sistem ekstraksi data.

Ekstrak:

- nama
- alamat
- nik
- nip
- tanggal

Output JSON valid.

Jangan menambahkan penjelasan.
Jangan menggunakan markdown.

Teks:

{text}
"""
```

Prompt harus:

- Pendek
- Jelas
- Deterministik

5.8 Integrasi Ollama dengan Python

Contoh dasar:

```
from ollama import chat

response = chat(
    model="qwen3",
    messages=[
        {
            "role": "user",
            "content": "Halo"
        }
    ]
)

print(response.message.content)
```

Ini adalah fondasi seluruh pipeline.

5.9 Fungsi Reusable NER

Tujuan utama:

```
extract_entities_ner()
```

Fungsi ini akan dipakai berulang kali.

Contoh:

```
result = extract_entities_ner(
    text,
    EntityDocument
)
```

Pendekatan ini membuat kode jauh lebih bersih.

5.10 Implementasi extract_entities_ner()

Contoh:

```
from ollama import chat

def extract_entities_ner(text):

    prompt = f"""
    Ekstrak entity dari teks berikut.

    Output JSON.

    {text}
    """

    response = chat(
        model="qwen3",
        messages=[
            {
                "role": "user",
                "content": prompt
            }
        ]
    )

    return response.message.content
```

Versi ini masih sederhana.

Nanti akan ditingkatkan.

5.11 Generic Schema Approach

Pendekatan modern menggunakan schema dinamis.

Contoh:

```
def extract_entities_ner(
    text,
    schema
):
    pass
```

Keuntungan:

Schema dapat berubah tanpa mengubah kode utama.

5.12 Structured Output dengan Pydantic

Contoh:

```
from pydantic import BaseModel

class InvoiceSchema(BaseModel):
    vendor: str
    tanggal: str
    total: float
```

Atau:

```
class KTPSchema(BaseModel):
    nama: str
    nik: str
    alamat: str
```

Pipeline menjadi sangat fleksibel.

5.13 Entity Standar

Mayoritas dokumen Indonesia membutuhkan:

- Nama
- Alamat
- Nomor KTP
- NIP
- NPWP
- Tanggal
- Nomor Surat
- Nomor Kontrak
- Produk
- Nominal

Entity ini dapat dijadikan fondasi awal.

5.14 Entity Custom

Setiap organisasi memiliki kebutuhan berbeda.

Contoh:

Instansi pendidikan:

```
{
  "nim": "",
  "program_studi": ""
}
```

Rumah sakit:

```
{
  "nomor_rekam_medis": "",
  "diagnosa": ""
}
```

Keunggulan LLM adalah kemampuan membuat entity baru tanpa pelatihan ulang.

5.15 Parsing JSON Output

Model menghasilkan teks.

Kita harus mengubahnya menjadi objek Python.

Contoh:

```
import json

data = json.loads(
    response
)
```

Jika gagal:

- Log error
- Jalankan retry

Jangan langsung menghentikan proses.

5.16 Error Handling

Masalah umum:

- JSON rusak
- Timeout
- Model berhenti
- Memori habis

Contoh:

```
try:
    data = json.loads(response)
except Exception:
    print("JSON Error")
```

Produksi tanpa error handling adalah bencana yang menunggu terjadi.

5.17 Retry Logic

Kadang model gagal menghasilkan JSON valid.

Solusi:

```
for i in range(3):

    try:
        return process()

    except Exception:
        pass
```

Prinsip:

- Retry maksimum 3 kali
- Simpan log kegagalan

5.18 Timeout Protection

Jangan biarkan proses menggantung.

Contoh:

```
timeout = 30
```

Jika melebihi batas:

```
FAILED
```

Lanjutkan ke dokumen berikutnya.

5.19 Confidence Scoring

Sebagian model tidak memberikan skor.

Kita dapat membuat estimasi sederhana.

Contoh:

```
confidence = 0.95
```

berdasarkan:

- Validitas JSON
- Kelengkapan field
- Validasi bisnis

5.20 Validasi NIK

Contoh:

```
if len(nik) != 16:  
    raise ValueError
```

Validasi sederhana sering mengurangi kesalahan secara drastis.

5.21 Logging Produksi

Instal:

```
pip install loguru
```

Contoh:

```
from loguru import logger

logger.info(
    "NER Success"
)
```

Simpan log:

```
logs/app.log
```

5.22 Contoh Log Produksi

```
2026-05-01
OCR Success
```

```
2026-05-01
NER Success
```

```
2026-05-01
INSERT Success
```

Administrator Linux harus membiasakan diri membaca log.

5.23 Koneksi MariaDB

File:

```
database.py
```

Contoh:

```
import pymysql

conn = pymysql.connect(
    host="localhost",
    user="neruser",
    password="passwordku",
    database="nerdb"
)
```

5.24 Membuat Tabel Entity

Contoh:

```
CREATE TABLE entities (  
    id INT AUTO_INCREMENT,  
    nama VARCHAR(255),  
    nik VARCHAR(50),  
    alamat TEXT,  
    PRIMARY KEY(id)  
);
```

Struktur dapat diperluas sesuai kebutuhan.

5.25 Generate Query INSERT Otomatis

Contoh:

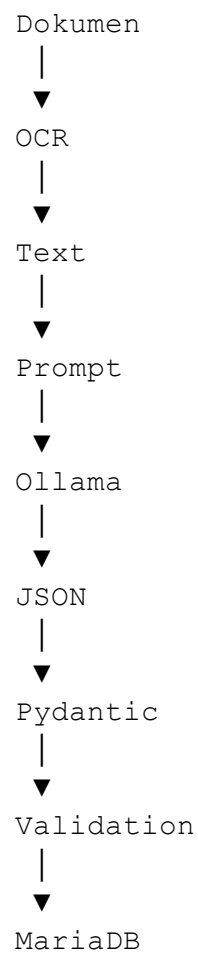
```
sql = ""  
INSERT INTO entities  
(  
    nama,  
    nik,  
    alamat  
)  
VALUES  
(  
    %s,  
    %s,  
    %s  
)  
""
```

Eksekusi:

```
cursor.execute(  
    sql,  
    (  
        data["nama"],  
        data["nik"],  
        data["alamat"]  
    )  
)
```

5.26 Pipeline Lengkap

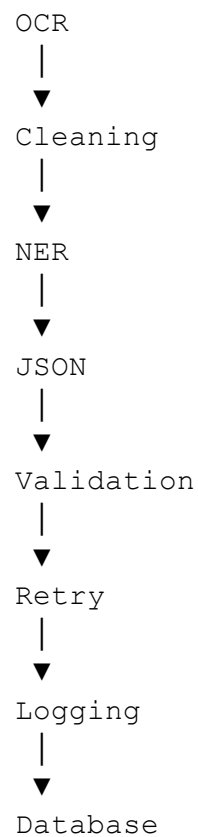
Alur akhirnya:



Inilah fondasi seluruh proyek pada bab 6–8.

5.27 Pipeline Produksi yang Direkomendasikan

Tambahkan:



Pipeline ini jauh lebih stabil dibanding contoh AI sederhana yang banyak beredar.

5.28 Checklist Akhir Bab

Pastikan pembaca mampu:

- Membuat schema Pydantic
- Membuat prompt produksi
- Mengakses Ollama dari Python
- Membuat fungsi reusable
- Menghasilkan JSON valid
- Membuat retry logic
- Membuat logging
- Menyimpan hasil ke MariaDB

Jika seluruh poin tercapai maka **core engine OCR + NER** sudah siap digunakan.

5.29 Penutup

Bab ini mengubah konsep menjadi sistem nyata. OCR menghasilkan teks, Ollama mengekstrak entitas, Pydantic memvalidasi struktur, dan MariaDB menyimpan hasil akhir. Kombinasi ini membentuk fondasi platform ekstraksi dokumen modern yang **mandiri, dapat diaudit, mudah dikembangkan, dan siap produksi**.

Pada bab berikutnya kita mulai memasuki proyek nyata pertama, yaitu **digitalisasi dokumen administrasi Indonesia**, termasuk KTP, surat resmi, formulir, dan ijazah, menggunakan pipeline OCR + NER yang telah dibangun pada bab ini.

BAB 6 - Proyek Hands-on 1 – Dokumen Administrasi Indonesia

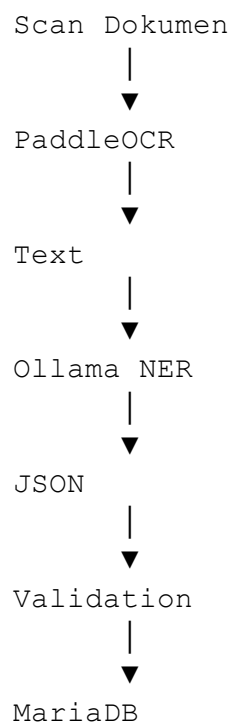
6.1 Dari Konsep Menjadi Solusi Nyata

Pada bab sebelumnya kita telah membangun:

- Ubuntu 26.04
- PaddleOCR
- Ollama
- Python
- Pydantic
- MariaDB

Kini saatnya menggabungkan semuanya.

Target proyek:



Output akhir bukan teks.

Output akhir adalah data terstruktur.

6.2 Studi Kasus Dokumen Indonesia

Dokumen yang akan diproses:

KTP

Ekstraksi:

- NIK
- Nama
- Tempat lahir
- Tanggal lahir
- Alamat
- Agama
- Pekerjaan

Surat Resmi

Ekstraksi:

- Nomor surat
- Tanggal
- Instansi
- Penandatanganan

Ijazah

Ekstraksi:

- Nama
- Nomor ijazah
- Program studi
- Tahun lulus

Sertifikat

Ekstraksi:

- Nama peserta
- Nomor sertifikat
- Tanggal terbit

6.3 Struktur Folder Proyek

Struktur yang direkomendasikan:

```
project-admin/  
├── input/  
│   ├── ktp/  
│   ├── surat/  
│   └── ijazah/  
├── output/  
├── logs/  
├── database/  
├── app/  
│   ├── ocr.py  
│   ├── ner.py  
│   ├── database.py  
│   ├── schemas.py  
│   └── postprocess.py  
└── main.py
```

Pendekatan ini memudahkan pengelolaan ribuan dokumen.

6.4 Alur Kerja Sistem

Langkah yang dilakukan:

Tahap 1

Membaca gambar.

Tahap 2

Menjalankan OCR.

Tahap 3

Membersihkan teks.

Tahap 4

Menjalankan NER.

Tahap 5

Validasi.

Tahap 6

Menyimpan ke database.

6.5 Membuat Schema KTP

File:

`app/schemas.py`

Contoh:

```
from pydantic import BaseModel

class KTPSchema(BaseModel):
    nik: str | None = None
    nama: str | None = None
    tempat_lahir: str | None = None
    tanggal_lahir: str | None = None
    alamat: str | None = None
    agama: str | None = None
    pekerjaan: str | None = None
```

Schema menjadi fondasi validasi.

6.6 Membuat Schema Surat

Contoh:

```
class SuratSchema(BaseModel):
    nomor_surat: str | None = None
    tanggal: str | None = None
    instansi: str | None = None
    penandatangan: str | None = None
```

6.7 Membuat Schema Ijazah

Contoh:

```
class IjazahSchema(BaseModel):
    nama: str | None = None
    nomor_ijazah: str | None = None
    program_studi: str | None = None
    tahun_lulus: str | None = None
```

6.8 OCR dengan PaddleOCR

File:

```
app/ocr.py
```

Contoh:

```
from paddleocr import PaddleOCR

ocr = PaddleOCR(
    use_angle_cls=True,
    lang="en"
)
```

Fungsi OCR:

```
def extract_text(image):
    result = ocr.ocr(image)
    text = ""
    for page in result:
        for line in page:
            text += line[1][0] + "\n"
    return text
```

6.9 Menguji OCR

Contoh:

```
text = extract_text(  
    "ktp.jpg"  
)  
  
print(text)
```

Output:

```
NIK  
3173012345678901
```

Nama

```
Budi Santoso
```

Tahap berikutnya adalah NER.

6.10 Prompt NER KTP

Contoh:

```
PROMPT_KTP = ""  
Ekstrak:
```

```
- nik  
- nama  
- tempat_lahir  
- tanggal_lahir  
- alamat  
- agama  
- pekerjaan
```

Output JSON valid.

Teks:

```
{text}  
""
```

Prompt harus sederhana dan spesifik.

6.11 Menjalankan Ollama

File:

app/ner.py

Contoh:

```
from ollama import chat

def run_ner(prompt):

    response = chat(
        model="qwen3",
        messages=[
            {
                "role": "user",
                "content": prompt
            }
        ]
    )

    return response.message.content
```

6.12 Pipeline OCR + NER

Contoh:

```
text = extract_text(
    "ktp.jpg"
)

prompt = PROMPT_KTP.format(
    text=text
)

result = run_ner(prompt)
```

Output:

```
{
    "nik": "3173012345678901",
    "nama": "Budi Santoso"
}
```

6.13 Custom Entity Indonesia

Keunggulan LLM adalah mampu menangani entity lokal.

Contoh:

NIK

```
{
  "nik": "3173xxxxxxxxxxxxx"
}
```

NIP

```
{
  "nip": "198104052005011001"
}
```

NPWP

```
{
  "npwp": "12.345.678.9-000.000"
}
```

Nomor Surat

```
{
  "nomor_surat": "123/ABC/2026"
}
```

Model klasik sering kesulitan mengenali pola tersebut.

6.14 Normalisasi Data

Masalah umum:

```
{
  "nama": "BUDI SANTOSO"
}
```

atau

```
{
  "nama": "budi santoso"
}
```

Normalisasi:

```
nama = nama.title()
```

Hasil:

```
{
  "nama": "Budi Santoso"
}
```

Lebih konsisten.

6.15 Normalisasi Tanggal

Sering muncul:

01 Januari 2026

ubah menjadi:

2026-01-01

Format standar memudahkan analitik.

6.16 Normalisasi Nomor

Contoh:

Rp 1.500.000,-

ubah menjadi:

1500000

Lebih mudah diproses.

6.17 Deduplikasi Data

Masalah nyata:

Dokumen sama diproses dua kali.

Solusi sederhana:

```
SELECT nik
FROM ktp
WHERE nik=%s
```

Jika sudah ada:

```
SKIP
```

6.18 Validasi NIK

Contoh:

```
if len(nik) != 16:

    raise ValueError(
        "NIK tidak valid"
    )
```

Jangan langsung percaya hasil OCR.

6.19 Validasi Nomor Surat

Contoh:

```
if "/" not in nomor_surat:

    raise ValueError
```

Aturan bisnis sederhana sering sangat efektif.

6.20 Membuat Database

Masuk:

```
mysql -u root -p
```

Buat database:

```
CREATE DATABASE administrasi;
```

6.21 Tabel KTP

Contoh:

```
CREATE TABLE ktp (
    id INT AUTO_INCREMENT,
    nik VARCHAR(30),
    nama VARCHAR(255),
    alamat TEXT,
    agama VARCHAR(100),
    pekerjaan VARCHAR(255),
    PRIMARY KEY(id),
    UNIQUE(nik)
);
```

6.22 Tabel Surat

Contoh:

```
CREATE TABLE surat (  
    id INT AUTO_INCREMENT,  
    nomor_surat VARCHAR(255),  
    tanggal DATE,  
    instansi VARCHAR(255),  
    penandatanganan VARCHAR(255),  
    PRIMARY KEY(id)  
);
```

6.23 Tabel Ijazah

Contoh:

```
CREATE TABLE ijazah (  
    id INT AUTO_INCREMENT,  
    nomor_ijazah VARCHAR(255),  
    nama VARCHAR(255),  
    program_studi VARCHAR(255),  
    tahun_lulus VARCHAR(10),  
    PRIMARY KEY(id)  
);
```

6.24 Insert Otomatis

Contoh:

```
cursor.execute(
    """
    INSERT INTO ktp
    (
        nik,
        nama,
        alamat
    )
    VALUES
    (
        %s,
        %s,
        %s
    )
    """,
    (
        data["nik"],
        data["nama"],
        data["alamat"]
    )
)
```

6.25 Batch Processing

Direktori:

```
input/ktp/
```

Contoh:

```
from pathlib import Path

for file in Path(
    "input/ktp"
).glob("*.*jpg"):

    process(file)
```

Satu perintah dapat memproses ratusan dokumen.

6.26 Logging Produksi

Contoh:

```
logger.info(  
    f"{file} sukses"  
)
```

Hasil:

```
2026-06-01  
ktp001.jpg sukses
```

```
2026-06-01  
ktp002.jpg sukses
```

6.27 Script CLI Lengkap

Contoh:

```
python main.py \  
--type ktp \  
--input input/ktp
```

Atau:

```
python main.py \  
--type surat \  
--input input/surat
```

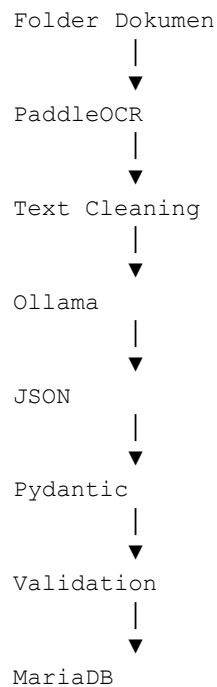
Atau:

```
python main.py \  
--type ijazah \  
--input input/ijazah
```

Pendekatan CLI sangat disukai administrator Linux karena mudah di otomasi menggunakan *cron*, *systemd timer*, maupun *shell script*.

6.28 Arsitektur Produksi

Implementasi akhir:



Arsitektur ini dapat digunakan untuk:

- Sekolah
- Kampus
- Pemerintah daerah
- Rumah sakit
- Perusahaan

6.29 Checklist Akhir Bab

Pastikan pembaca mampu:

- Membaca KTP dengan OCR
- Membaca surat resmi
- Membaca ijazah
- Membuat schema domain Indonesia
- Menjalankan NER menggunakan Ollama
- Melakukan normalisasi data
- Melakukan deduplikasi
- Menyimpan ke MariaDB
- Memproses banyak dokumen sekaligus

Jika seluruh poin tercapai maka pembaca telah berhasil membangun sistem digitalisasi dokumen administrasi Indonesia secara end-to-end.

6.30 Penutup

Bab ini menunjukkan bahwa kombinasi **PaddleOCR, Ollama, Python, Pydantic, dan MariaDB** mampu mengubah dokumen administrasi Indonesia menjadi data terstruktur yang siap digunakan. Nilai sebenarnya bukan pada OCR atau LLM secara terpisah, melainkan pada kemampuan mengintegrasikan seluruh komponen menjadi pipeline yang **otomatis, dapat diaudit, mudah diperluas, dan siap produksi**.

Pada bab berikutnya kita akan melangkah ke dokumen yang lebih kompleks, yaitu **invoice, kwitansi, kontrak, dan dokumen bisnis**, yang melibatkan ekstraksi tabel, nominal transaksi, relasi antar entitas, serta validasi aturan bisnis yang jauh lebih menantang dibanding dokumen administrasi biasa.

BAB 7 - Proyek Hands-on 2 – Dokumen Bisnis dan Keuangan

7.1 Mengapa Dokumen Bisnis Lebih Sulit?

Pada dokumen administrasi, sebagian besar informasi bersifat sederhana.

Contoh:

- Nama
- NIK
- Alamat
- Nomor Surat

Pada dokumen bisnis situasinya berbeda.

Satu invoice dapat memiliki:

- Vendor
- Nomor invoice
- Tanggal
- Banyak item
- Harga satuan
- Pajak
- Diskon
- Total transaksi

Kesalahan OCR satu angka saja dapat menyebabkan:

- Salah pembayaran
- Salah pencatatan
- Salah audit

Karena itu diperlukan validasi tambahan.

7.2 Jenis Dokumen yang Akan Diproses

Fokus proyek:

Invoice

Ekstraksi:

- Vendor
- Nomor invoice
- Tanggal
- Daftar item
- Pajak
- Total

Kwitansi

Ekstraksi:

- Nomor kwitansi
- Penerima
- Nominal
- Tanggal

Purchase Order

Ekstraksi:

- Vendor
- Nomor PO
- Produk
- Jumlah

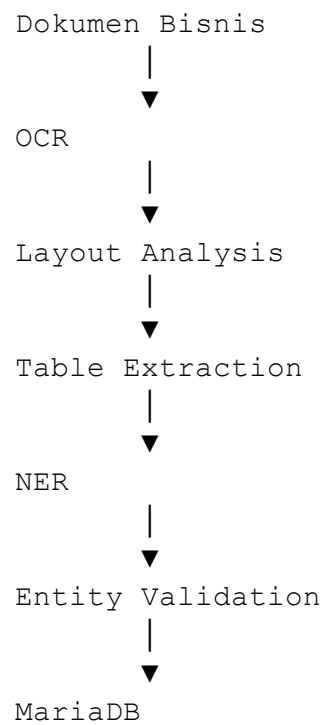
Kontrak

Ekstraksi:

- Nomor kontrak
- Pihak pertama
- Pihak kedua
- Nilai kontrak
- Tanggal berlaku

7.3 Arsitektur Sistem

Pipeline yang digunakan:



Tahapan baru yang muncul adalah:

- *Table Extraction*
- *Business Validation*

7.4 Struktur Folder Proyek

Direkomendasikan:

```
project-finance/  
├── input/  
│   ├── invoice/  
│   ├── kontrak/  
│   └── kwitansi/  
├── output/  
├── logs/  
├── app/  
│   ├── ocr.py  
│   ├── table.py  
│   ├── ner.py  
│   ├── validation.py  
│   ├── database.py  
│   └── schemas.py  
└── main.py
```

Struktur ini mudah diperluas untuk ribuan dokumen.

7.5 Tantangan OCR pada Invoice

OCR sering gagal pada:

- Tabel
- Garis tabel
- Nominal uang
- Simbol mata uang
- Kolom multi-item

Contoh:

Produk A	10	250000
Produk B	15	100000

Jika OCR salah membaca satu angka:

250000

menjadi:

350000

maka seluruh laporan keuangan menjadi salah.

7.6 Schema Invoice

File:

`app/schemas.py`

Contoh:

```
from pydantic import BaseModel

class InvoiceSchema(BaseModel):
    vendor: str | None = None
    nomor_invoice: str | None = None
    tanggal: str | None = None
    subtotal: float | None = None
    pajak: float | None = None
    total: float | None = None
```

7.7 Schema Item Invoice

Setiap invoice dapat memiliki banyak item.

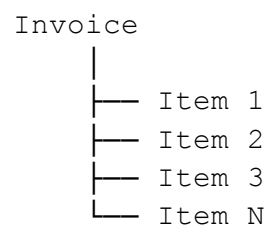
Contoh:

```
from pydantic import BaseModel

class InvoiceItem(BaseModel):
    nama_produk: str
    jumlah: int
    harga_satuan: float
    total_item: float
```

7.8 Relasi Header dan Detail

Secara konseptual:



Pendekatan ini lebih sesuai dengan praktik basis data produksi.

7.9 Table Extraction

Tahapan baru pada dokumen bisnis adalah ekstraksi tabel.

Contoh tabel:

Produk	Qty	Harga
Router	2	500000
Switch	1	1000000

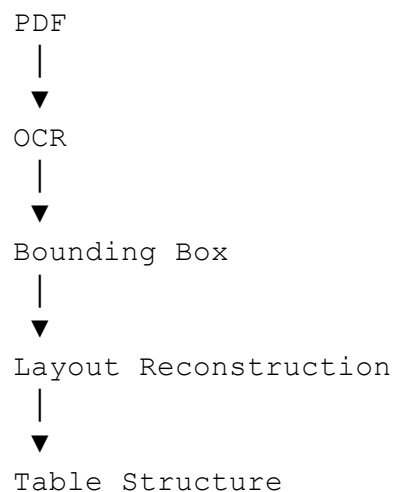
OCR biasa sering gagal mempertahankan struktur ini.

Karena itu perlu dilakukan:

- Layout detection
- Cell detection
- Row reconstruction

7.10 OCR dan Layout Analysis

Pendekatan modern:



Tujuan:

Mengembalikan struktur dokumen yang hilang.

7.11 Prompt Invoice

Contoh:

```
PROMPT_INVOICE = ""  
Ekstrak:  
  
- vendor  
- nomor_invoice  
- tanggal  
- subtotal  
- pajak  
- total  
  
Keluarkan JSON valid.  
  
Teks:  
  
{text}  
""
```

Prompt harus fokus.
Jangan terlalu panjang.

7.12 Prompt Kontrak

Contoh:

```
PROMPT_KONTRAK = ""  
Ekstrak:  
  
- nomor_kontrak  
- pihak_pertama  
- pihak_kedua  
- nilai_kontrak  
- tanggal_mulai  
- tanggal_selesai  
  
Output JSON valid.  
""
```

7.13 Entity Relationship Extraction

Dokumen bisnis tidak hanya membutuhkan entity.
Kita juga perlu relasi.

Contoh:

PT ABC menjual Router kepada PT XYZ

Entity:

```
{
  "vendor": "PT ABC",
  "customer": "PT XYZ",
  "produk": "Router"
}
```

Relasi:

```
PT ABC
  |
MENJUAL
  |
PT XYZ
```

Informasi ini sangat berguna untuk analitik.

7.14 Relasi dalam Kontrak

Contoh:

PT A dan PT B menandatangani kontrak.

Hasil:

```
{
  "pihak_1": "PT A",
  "pihak_2": "PT B"
}
```

Relasi ini dapat digunakan untuk membangun *knowledge graph* organisasi.

7.15 Normalisasi Nominal

Masalah umum:

Rp 1.500.000,-

atau

1,500,000.00

Semuanya harus diubah menjadi:

1500000

Agar mudah dianalisis.

7.16 Validasi Nominal

Contoh:

```
if total < 0:
    raise ValueError
```

Nilai negatif pada invoice biasanya merupakan kesalahan.

7.17 Validasi Tanggal

Contoh:

```
from datetime import datetime

datetime.strptime(
    tanggal,
    "%Y-%m-%d"
)
```

Jika gagal:

REJECT

7.18 Validasi Perhitungan Invoice

Aturan bisnis sederhana:

```
Subtotal
+
Pajak
=
Total
```

Jika tidak sesuai:

```
FLAG
```

Contoh:

```
1000000
+
110000
=
1110000
```

Harus konsisten.

7.19 Validasi Item Invoice

Contoh:

```
Qty × Harga
=
Total Item
```

Jika:

```
10 × 50000
```

hasilnya bukan:

```
500000
```

maka ada kesalahan.

7.20 Database Header Invoice

Contoh:

```
CREATE TABLE invoice (  
    id INT AUTO_INCREMENT,  
    vendor VARCHAR(255),  
    nomor_invoice VARCHAR(255),  
    tanggal DATE,  
    subtotal DECIMAL(18,2),  
    pajak DECIMAL(18,2),  
    total DECIMAL(18,2),  
    PRIMARY KEY(id)  
);
```

7.21 Database Detail Invoice

Contoh:

```
CREATE TABLE invoice_item (  
    id INT AUTO_INCREMENT,  
    invoice_id INT,  
    nama_produk VARCHAR(255),  
    qty INT,  
    harga DECIMAL(18,2),  
    total DECIMAL(18,2),  
    PRIMARY KEY(id)  
);
```

7.22 Konsep Transaction Safety

Masalah:

Header berhasil disimpan.

Detail gagal.

Akibatnya database rusak.

Karena itu gunakan transaksi.

7.23 Contoh Transaction

```
conn.begin()
```

Simpan:

```
invoice
```

Simpan:

```
invoice_item
```

Jika berhasil:

```
conn.commit()
```

Jika gagal:

```
conn.rollback()
```

7.24 Mengapa Transaction Penting

Tanpa transaksi:

Invoice tersimpan

Item gagal

Data menjadi tidak lengkap.

Dengan transaksi:

Semua berhasil

atau

Semua dibatalkan

Jauh lebih aman.

7.25 Bulk Processing

Tujuan:

Memproses ribuan dokumen.

Contoh:

```
from pathlib import Path

for file in Path(
    "input/invoice"
).glob("*.pdf"):

    process_invoice(file)
```

7.26 Batch Insert

Daripada:

```
INSERT
```

satu per satu.

Gunakan:

```
executemany()
```

Keuntungan:

- Lebih cepat
- Lebih hemat sumber daya

7.27 Logging Audit

Contoh:

```
2026-06-01
invoice001.pdf
SUCCESS

2026-06-01
invoice002.pdf
FAILED
```

Log audit sangat penting dalam lingkungan keuangan.

7.28 Dashboard Statistik Sederhana

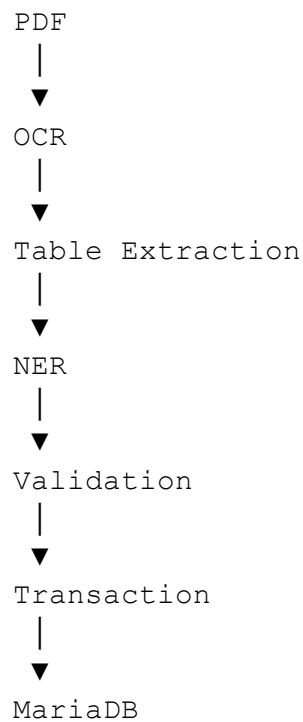
Informasi yang dapat dihitung:

- Jumlah invoice
- Total transaksi
- Vendor terbanyak
- Nilai kontrak terbesar

Semua dapat diperoleh dari hasil ekstraksi NER.

7.29 Arsitektur Produksi

Implementasi akhir:



Pendekatan ini cukup kuat untuk banyak kebutuhan bisnis.

7.30 Checklist Akhir Bab

Pastikan pembaca mampu:

- Mengekstrak invoice
- Mengekstrak kwitansi
- Mengekstrak kontrak
- Mengekstrak tabel
- Membuat relasi entity
- Melakukan validasi bisnis
- Menggunakan transaksi database
- Melakukan batch processing
- Melakukan audit logging

Jika seluruh poin tercapai maka pembaca telah berhasil membangun sistem OCR + NER untuk dokumen bisnis dan keuangan yang mendekati kebutuhan produksi.

7.31 Penutup

Dokumen bisnis jauh lebih kompleks dibanding dokumen administrasi. Tantangan utama bukan lagi membaca teks, tetapi memahami struktur transaksi, relasi antar entitas, dan memastikan konsistensi aturan bisnis. Dengan kombinasi **PaddleOCR, Ollama, Python, Pydantic, dan MariaDB**, organisasi dapat membangun sistem ekstraksi dokumen keuangan yang **akurat, dapat diaudit, aman, dan sepenuhnya berjalan di infrastruktur lokal**.

Pada bab berikutnya kita akan masuk ke skenario yang lebih menantang lagi, yaitu **arsip sejarah, dokumen medis, dan pemrosesan skala besar**, termasuk ribuan dokumen, *batch processing, queue management*, optimasi GPU, serta teknik peningkatan akurasi menggunakan model yang lebih spesifik terhadap domain tertentu.

BAB 8 - Proyek Hands-on 3 – Arsip Sejarah, Medis, dan Skala Besar

8.1 Ketika OCR Masuk Dunia Nyata

Pada bab sebelumnya kita memproses:

- KTP
- Surat
- Invoice
- Kontrak

Dokumen relatif bersih.

Pada dunia nyata situasinya jauh lebih sulit.

Contoh:

- Arsip tahun 1950-an
- Dokumen kolonial
- Surat tulisan mesin tik
- Rekam medis lama
- Dokumen hasil fotokopi berkali-kali
- Arsip lembab dan rusak

Masalah terbesar bukan lagi NER.

Masalah utama adalah kualitas dokumen.

8.2 Karakteristik Arsip Sejarah

Dokumen sejarah sering memiliki:

Noise Tinggi

- Bercak
- Jamur
- Noda tinta

Kertas Rusak

- Sobek
- Kusut
- Menguning

Font Kuno

- Mesin tik
- Tulisan tangan

Layout Tidak Konsisten

Setiap dokumen bisa berbeda.

OCR modern masih kesulitan menghadapi kondisi tersebut.

8.3 Karakteristik Dokumen Medis

Dokumen medis memiliki tantangan berbeda.

Contoh entity:

- Nomor rekam medis
- Nama pasien
- Diagnosa
- Tindakan medis
- Obat
- Dokter

Masalah utama:

Istilah Medis

Sangat spesifik.

Singkatan

Contoh:

HT
DM
CKD

Tulisan Tidak Konsisten

Banyak rumah sakit masih memiliki arsip lama.

8.4 Mengapa Domain-Specific NER Penting

NER umum mampu mengenali:

- Nama
- Lokasi
- Organisasi

Tetapi belum tentu mampu mengenali:

Gagal Ginjal Kronis

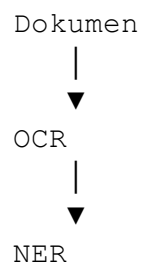
sebagai:

```
{  
  "diagnosa": "Gagal Ginjal Kronis"  
}
```

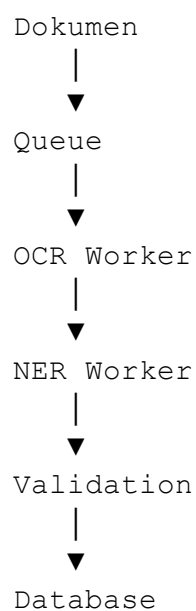
Karena itu diperlukan pendekatan khusus.

8.5 Arsitektur Produksi Skala Besar

Pada skala kecil:



Pada skala besar:



Pendekatan ini lebih stabil.

8.6 Struktur Folder Produksi

Direkomendasikan:

```
project-scale/
├── incoming/
├── processing/
├── completed/
├── failed/
├── queue/
├── cache/
├── vectorstore/
├── logs/
└── app/
```

Tujuan:

- Mudah dipantau
- Mudah dipulihkan
- Mudah diaudit

8.7 Domain-Specific Schema

Contoh schema medis:

```
from pydantic import BaseModel

class MedicalRecord(BaseModel):
    nomor_rm: str | None = None
    nama_pasien: str | None = None
    diagnosa: str | None = None
    obat: str | None = None
    dokter: str | None = None
```

8.8 Schema Arsip Sejarah

Contoh:

```
class HistoricalDocument(BaseModel):  
    nomor_arsip: str | None = None  
    tokoh: list[str] = []  
    lokasi: list[str] = []  
    tanggal: str | None = None  
    peristiwa: str | None = None
```

Schema dapat disesuaikan sesuai kebutuhan.

8.9 Prompt Domain Khusus

Contoh arsip sejarah:

Ekstrak:

- tokoh
- lokasi
- tanggal
- peristiwa

Output JSON valid.

Contoh medis:

Ekstrak:

- diagnosa
- obat
- dokter

Output JSON valid.

Semakin spesifik domain, semakin baik hasilnya.

8.10 Pipeline Pemrosesan Massal

Target:

10.000+
100.000+
1.000.000+
dokumen

Pendekatan satu file satu proses tidak lagi cukup.

Diperlukan sistem antrian.

8.11 Konsep Queue

Folder:

`incoming/`

berisi dokumen baru.

Worker membaca:

`incoming/`

dan memindahkan ke:

`processing/`

Jika berhasil:

`completed/`

Jika gagal:

`failed/`

Pendekatan sederhana tetapi sangat efektif.

8.12 Worker OCR

Contoh:

```
while True:
    file = get_next_file()
    process_ocr(file)
```

Worker hanya fokus pada OCR.

8.13 Worker NER

Contoh:

```
while True:
    document = get_next_text()
    process_ner(document)
```

Pemisahan worker meningkatkan stabilitas.

8.14 Paralel Processing

Satu proses:

1 Dokumen

beberapa proses:

Worker 1
Worker 2
Worker 3
Worker 4

Keuntungan:

- Lebih cepat
- Memanfaatkan CPU maksimal

8.15 GPU Scheduling

Masalah umum:

GPU 0

Penuh

GPU 1

Kosong

Akibatnya pemrosesan tidak optimal.

Strategi sederhana:

Round Robin

Distribusikan pekerjaan secara merata.

8.16 Monitoring GPU

Periksa penggunaan GPU:

```
nvidia-smi
```

Monitoring kontinu:

```
watch -n 1 nvidia-smi
```

Atau:

```
nvidia-smi
```

Administrator Linux wajib memahami kondisi GPU.

8.17 Batch Processing

Daripada:

```
1 dokumen
```

proses:

```
100 dokumen
```

```
sekaligus.
```

Contoh:

```
batch_size = 100
```

Keuntungan:

- Mengurangi overhead
- Meningkatkan throughput

8.18 Mengukur Throughput

Contoh:

1000 dokumen

selesai dalam:

10 menit

Maka throughput:

$\text{Throughput} = 1000 / 10 = 100 \text{ dokumen/menit}$

Metrik ini penting untuk produksi.

8.19 Caching Hasil OCR

Masalah:

Dokumen sama diproses berulang.

Solusi:

Cache OCR

Jika hasil sudah ada:

Jangan OCR ulang

Keuntungan:

- Lebih cepat
- Hemat GPU
- Hemat listrik

8.20 Caching Hasil NER

Konsep sama.

Simpan:

```
{  
  "hash": "abc123",  
  "result": "..."  
}
```

Jika hash ditemukan:

RETURN CACHE

Tidak perlu inferensi ulang.

8.21 Menggunakan Hash Dokumen

Contoh:

```
import hashlib

hash_file = hashlib.sha256(
    data
).hexdigest()
```

Hash menjadi identitas unik dokumen.

8.22 Fine-Tuning Ringan dengan LoRA

Sebagian organisasi memiliki dokumen khusus.

Contoh:

- Rekam medis
- Arsip pengadilan
- Dokumen pertanahan

Model umum mungkin kurang optimal.

Solusi:

LoRA Fine-Tuning.

8.23 Kapan LoRA Dibutuhkan?

Tidak semua proyek memerlukan fine tuning.

Lakukan hanya jika:

- Dataset cukup besar
- Domain sangat spesifik
- Akurasi masih rendah

Jika akurasi sudah memadai:

Jangan melakukan fine tuning.

8.24 Dataset Fine-Tuning

Format sederhana:

```
{  
  "input": "dokumen",  
  "output": "entity"  
}
```

Semakin bersih dataset:
Semakin baik hasil model.

8.25 Risiko Fine-Tuning

Kesalahan umum:

Dataset Terlalu Sedikit

Model justru memburuk.

Data Kotor

Model belajar kesalahan.

Overfitting

Model hanya pintar pada data pelatihan.

8.26 Vector Search untuk Arsip

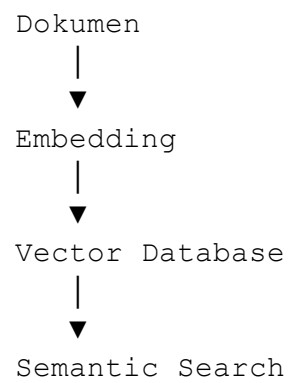
Setelah entity diekstrak.
Kita dapat membuat pencarian semantik.

Contoh:

Cari semua arsip
tentang kemerdekaan

tanpa mengetahui kata persisnya.

8.27 Arsitektur Vector Search



Pencarian menjadi jauh lebih cerdas dibanding kata kunci biasa.

8.28 Embedding Lokal

Model embedding ringan dapat berjalan lokal.

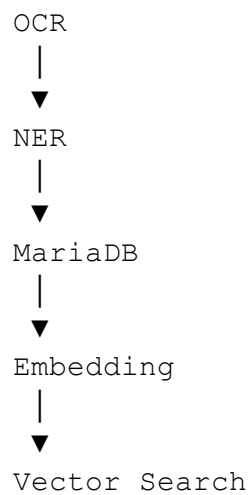
Output:

```
[  
  0.123,  
  0.456,  
  ...  
]
```

Vektor ini menjadi dasar pencarian semantik.

8.29 Integrasi NER dan Vector Search

Arsitektur lengkap:



Dokumen tidak hanya tersimpan.

Dokumen menjadi dapat dieksplorasi.

8.30 Analitik Skala Besar

Contoh:

Arsip Sejarah

- Tokoh paling sering muncul
- Lokasi paling sering disebut
- Periode waktu dominan

Medis

- Diagnosa terbanyak
- Obat paling sering diresepkan
- Distribusi penyakit

NER membuka peluang analitik yang sangat besar.

8.31 Disaster Recovery

Selalu siapkan:

Backup Database

`mysqldump`

Backup Cache

`tar`

Backup Model

`/opt/models`

Jangan menunggu sistem gagal.

8.32 Audit dan Kepatuhan

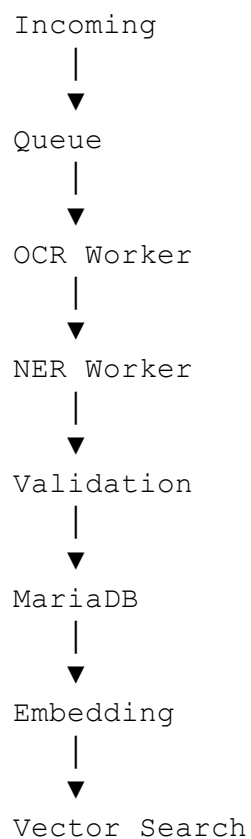
Catat:

- Dokumen diproses kapan
- Oleh siapa
- Menggunakan model apa
- Hasil validasi

Audit trail sangat penting pada:

- Pemerintahan
- Rumah sakit
- Arsip nasional

8.33 Arsitektur Produksi Lengkap



Inilah bentuk sistem yang mendekati produksi sesungguhnya.

8.34 Checklist Akhir Bab

Pastikan pembaca mampu:

- Menangani dokumen rusak
- Membuat schema domain khusus
- Menggunakan queue
- Menjalankan batch processing
- Mengelola GPU
- Membuat cache OCR
- Membuat cache NER
- Memahami LoRA
- Membangun vector search sederhana

Jika seluruh poin tercapai maka pembaca telah siap membangun platform OCR + NER skala besar.

8.35 Penutup

Tantangan terbesar dalam digitalisasi bukan membaca satu dokumen, melainkan mengelola jutaan dokumen yang berbeda kualitas, struktur, dan domain. Kombinasi **OCR modern, Ollama, Python, cache cerdas, queue processing, LoRA, dan vector search** memungkinkan organisasi membangun platform ekstraksi informasi yang **cepat, hemat biaya, sepenuhnya lokal, dan mampu berkembang mengikuti kebutuhan masa depan**.

Pada bab berikutnya kita akan membahas **optimasi performa, keamanan, deployment, monitoring, serta praktik terbaik produksi**, agar sistem OCR + NER yang telah dibangun benar-benar siap digunakan pada lingkungan operasional yang kritikal dan berkelanjutan.

BAB 9 - Optimasi, Keamanan, dan Produksi

9.1 Dari Laboratorium ke Produksi

Pada bab sebelumnya kita telah membangun:

- OCR
- NER
- Database
- Queue
- Cache
- Vector Search

Sistem sudah berjalan.

Namun belum tentu siap produksi.

Lingkungan produksi memiliki tantangan berbeda:

- Ribuan dokumen per hari
- Banyak pengguna
- Keterbatasan GPU
- Serangan siber
- Kegagalan perangkat keras
- Audit dan kepatuhan

Karena itu diperlukan optimasi dan pengamanan tambahan.

9.2 Filosofi Produksi

Prinsip utama:

Sistem yang stabil lebih berharga dibanding sistem yang hanya cepat.

Banyak proyek AI gagal karena:

- Tidak ada monitoring
- Tidak ada backup
- Tidak ada logging
- Tidak ada validasi

Produksi bukan hanya tentang model.

Produksi adalah tentang operasi.

9.3 Parameter yang Harus Dimonitor

Empat indikator utama:

Performa

- Latency
- Throughput

Infrastruktur

- CPU
- RAM
- GPU
- Storage

Akurasi

- CER
- WER
- NER Accuracy

Operasional

- Error Rate
- Success Rate

9.4 Monitoring CPU

Periksa CPU:

```
lscpu
```

Monitoring real-time:

```
htop
```

Atau:

```
top
```

Perhatikan:

- CPU Load
- Thread
- Bottleneck

9.5 Monitoring Memori

Periksa:

```
free -h
```

Detail:

```
vmstat 1
```

Jika RAM selalu penuh:

- Tambah RAM
- Gunakan model lebih kecil
- Tingkatkan cache

9.6 Monitoring GPU

Periksa:

```
nvidia-smi
```

Monitoring berkelanjutan:

```
watch -n 1 nvidia-smi
```

Atau:

```
nvidia-smi
```

Parameter penting:

- VRAM
- Utilisasi GPU
- Temperatur

9.7 Monitoring Storage

Periksa:

```
df -h
```

Periksa inode:

```
df -i
```

Banyak sistem gagal karena disk penuh.
Bukan karena model.

9.8 Mengukur Latency

Latency adalah waktu yang dibutuhkan untuk memproses satu dokumen.
Contoh:

```
import time
start = time.time()
process_document()
end = time.time()
print(end - start)
```

Target realistis:

Dokumen	Target
KTP	< 3 detik
Surat	< 5 detik
Invoice	< 10 detik
Kontrak	< 20 detik

9.9 Mengukur Throughput

Contoh:

500 dokumen

diproses dalam:

10 menit

Maka:

$\text{Throughput} = 500 / 10 = 50 \text{ dokumen/menit}$

Semakin tinggi semakin baik.

9.10 Mengukur Akurasi OCR

Gunakan:

CER

$$\text{CER} = (S+D+I) / N$$

Target:

< 5%

WER

$$\text{WER} = S+D+I / N$$

Target:

< 10%

9.11 Mengukur Akurasi NER

Hitung:

- Entity benar
- Entity salah
- Entity terlewat

Contoh:

100 entity

Benar:

95

Maka akurasi:

95%

Catat hasil secara berkala.

9.12 Mengapa Data Lokal Penting

Dokumen sering berisi:

- KTP
- Rekam medis
- Kontrak
- Data pelanggan

Mengirim data ke pihak luar meningkatkan risiko:

- Kebocoran data
- Pelanggaran regulasi
- Kehilangan kendali

Pendekatan lokal memberikan:

Privasi
Kepatuhan
Kontrol Penuh
Biaya Lebih Rendah

9.13 Enkripsi Penyimpanan

Direkomendasikan:

Enkripsi Disk

Melindungi data saat perangkat dicuri.

Enkripsi Backup

Melindungi data saat dipindahkan.

Enkripsi Database

Melindungi data sensitif.

Prinsip:

Data sensitif harus diasumsikan akan dicuri suatu hari nanti.

9.14 Hardening Server

Langkah minimum:

Update Berkala

```
sudo apt update  
sudo apt upgrade
```

Firewall

```
sudo ufw enable
```

SSH Key

Hindari autentikasi berbasis kata sandi jika memungkinkan.

Least Privilege

Jangan menjalankan aplikasi sebagai root.

9.15 Isolasi Layanan

Pisahkan:

```
OCR  
NER  
Database  
Vector Search
```

Jika satu komponen gagal:

Komponen lain tetap berjalan.

9.16 Logging Produksi

Contoh:

```
logger.info(  
    "OCR Success"  
)
```

Contoh log:

```
2026-06-01  
OCR Success
```

```
2026-06-01  
NER Success
```

```
2026-06-01  
INSERT Success
```

Log adalah sahabat administrator.

9.17 Audit Trail

Catat:

- Dokumen
- Waktu proses
- Model
- Pengguna
- Status

Contoh:

```
{  
  "dokumen": "invoice001.pdf",  
  "model": "qwen3",  
  "status": "success"  
}
```

Audit sangat penting pada:

- Pemerintah
- Kesehatan
- Keuangan

9.18 Deployment dengan Container

Container mempermudah:

- Instalasi
- Replikasi
- Migrasi
- Pemeliharaan

Struktur sederhana:

OCR Service
NER Service
MariaDB
Web Interface

berjalan pada container terpisah.

9.19 Contoh Dockerfile

```
FROM ubuntu:26.04

RUN apt update

RUN apt install -y \
python3 \
python3-pip

COPY . /app

WORKDIR /app

RUN pip install -r requirements.txt

CMD ["python3", "main.py"]
```

Pendekatan ini mempermudah distribusi aplikasi.

9.20 Menjalankan Sebagai Service

Aplikasi produksi sebaiknya berjalan sebagai service.

Contoh:

```
sudo systemctl enable ner.service
```

Start:

```
sudo systemctl start ner.service
```

Status:

```
systemctl status ner.service
```

9.21 Wrapper API dengan FastAPI

Mengubah pipeline menjadi layanan API.

Instalasi:

```
pip install fastapi uvicorn
```

Contoh:

```
from fastapi import FastAPI

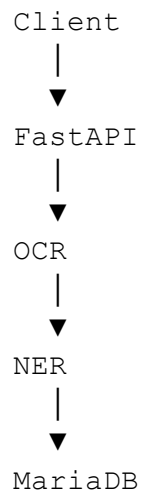
app = FastAPI()

@app.get("/")
def home():
    return {"status": "ok"}
```

Keuntungan:

- Mudah diintegrasikan
- Mendukung otomatisasi
- Cocok untuk produksi

9.22 Arsitektur API Produksi



Pendekatan ini lebih fleksibel dibanding aplikasi CLI saja.

9.23 Antarmuka Web

Administrator sering membutuhkan:

- Upload dokumen
- Monitoring
- Audit

Antarmuka web mempermudah operasional harian.

Namun API tetap menjadi fondasi utama.

9.24 Masalah OOM (*Out of Memory*)

Gejala:

`Killed`

atau:

`CUDA Out Of Memory`

Penyebab:

- Model terlalu besar
- Batch terlalu besar
- VRAM habis

Solusi:

- Gunakan model lebih kecil
- Kurangi batch
- Tambah RAM atau VRAM

9.25 Hallucination

Gejala:

Model menciptakan informasi yang tidak ada.

Contoh:

Dokumen tidak memiliki NIK.
Model membuat NIK sendiri.

Solusi:

- Schema ketat
- Pydantic
- Validasi bisnis
- Prompt sederhana

9.26 JSON Tidak Valid

Gejala:

```
json.decoder.JSONDecodeError
```

Solusi:

- Retry
- Validasi schema
- Kurangi kompleksitas prompt

9.27 OCR Buruk

Penyebab:

- Blur
- Noise
- Resolusi rendah

Solusi:

- Deskew
- Denoise
- CLAHE
- Rescan dokumen

Masalah OCR sering disalah artikan sebagai masalah LLM.

9.28 Masalah Multibahasa

Dokumen Indonesia sering berisi:

- Indonesia
- Inggris
- Arab
- Istilah teknis

Solusi:

- Gunakan model multilingual
- Tambahkan contoh pada prompt

9.29 Masalah Koneksi MariaDB

Gejala:

```
Connection Refused
```

Periksa:

```
systemctl status mariadb
```

Periksa port:

```
ss -tulpn
```

Periksa kredensial:

```
host  
user  
password  
database
```

9.30 Masalah Kinerja Menurun

Penyebab umum:

- Disk hampir penuh
- Log terlalu besar
- Cache tidak dibersihkan
- Terlalu banyak proses

Periksa secara berkala.

9.31 Backup Produksi

Database:

```
mysqldump
```

Konfigurasi:

```
tar
```

Model:

```
/opt/models
```

Backup harus otomatis.

Bukan manual.

9.32 High Availability Sederhana

Minimal:

- Backup harian
- Server cadangan
- Replikasi database

Tujuan:

Mengurangi *downtime*.

9.33 Checklist Produksi

Pastikan:

Infrastruktur

- Monitoring aktif
- Backup aktif

AI

- Latency terukur
- Akurasi tercatat

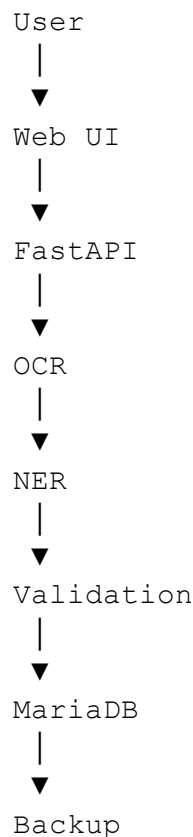
Keamanan

- Firewall aktif
- Enkripsi aktif

Operasional

- Logging aktif
- Audit trail aktif

9.34 Arsitektur Produksi Lengkap



Inilah bentuk sistem OCR + NER yang siap digunakan pada lingkungan operasional.

9.35 Penutup

Membangun OCR dan NER hanyalah langkah awal. Tantangan sesungguhnya adalah menjaga sistem tetap cepat, aman, akurat, dan dapat diandalkan selama bertahun-tahun. Organisasi yang berhasil bukanlah yang memiliki model terbesar, melainkan yang mampu mengelola infrastruktur, keamanan, monitoring, dan operasional secara disiplin.

Pada bab terakhir kita akan membahas **masa depan NER lokal, multimodal AI, Small Language Models, agentic workflow, komunitas open source Indonesia, serta peluang membangun kedaulatan AI berbasis infrastruktur sendiri**, sehingga pembaca tidak hanya mampu menggunakan teknologi, tetapi juga mampu berkontribusi mengembangkannya.

BAB 10 - Masa Depan NER Lokal dan Inspirasi

10.1 Kita Baru Memulai

Jika Anda telah mengikuti buku ini sampai bab terakhir, maka Anda telah berhasil membangun:

- OCR modern
- NER berbasis LLM
- Pipeline Python
- Basis data terstruktur
- Pemrosesan massal
- Caching
- Vector Search
- Deployment lokal

Ini bukan pencapaian kecil.

Mayoritas organisasi masih berada pada tahap:

```
PDF
↓
Disimpan
↓
Dilupakan
```

Sedangkan Anda telah mampu mengubah dokumen menjadi pengetahuan yang dapat dicari, dianalisis, dan dimanfaatkan.

Namun perjalanan belum selesai.

10.2 Evolusi OCR dan NER

Generasi pertama:

OCR
↓
Text

Generasi kedua:

OCR
↓
NER
↓
Database

Generasi ketiga:

OCR
↓
NER
↓
Knowledge Graph
↓
AI Assistant

Generasi keempat yang mulai muncul:

OCR
↓
NER
↓
Knowledge Graph
↓
Agent
↓
Action

Dokumen tidak hanya dibaca.

Dokumen mulai dipahami.

10.3 Tren 2026–2027

Beberapa tren mulai terlihat jelas.

Multimodal AI
Small Language Model
Agentic Workflow
Local AI
Edge AI
Domain-Specific AI

Kelima tren ini akan sangat mempengaruhi dunia OCR dan NER.

10.4 Multimodal Model

Dulu:

Teks

Sekarang:

Teks
+
Gambar
+
Tabel
+
Diagram
+
Formulir

dipahami secara bersamaan.

Model mulai mampu memahami:

- Layout
- Grafik
- Stempel
- Tanda tangan
- Diagram teknis

dalam satu proses.

Batas antara OCR dan NER semakin menghilang.

10.5 Dari OCR ke Document Understanding

Tujuan masa depan bukan lagi:
Membaca karakter

Tetapi:
Memahami dokumen

Contoh:
Dokumen kontrak.

AI tidak hanya mengambil:

- Nomor kontrak
- Nama pihak

Tetapi juga memahami:

- Risiko
- Kewajiban
- Tenggat waktu

Inilah arah industri saat ini.

10.6 Small Language Models (SLM)

Beberapa tahun terakhir fokus dunia AI adalah model besar.

Namun tren mulai bergeser.

Model kecil semakin menarik karena:

Lebih Murah
Lebih Cepat
Lebih Hemat Energi
Lebih Mudah Dioperasikan
Cocok untuk Infrastruktur Lokal

Banyak organisasi sebenarnya tidak membutuhkan model raksasa.

Mereka hanya membutuhkan model yang tepat.

10.7 Mengapa SLM Penting untuk Indonesia

Realitas lapangan:

- Sekolah
- Kampus
- Pemerintah daerah
- Rumah sakit
- UMKM

sering memiliki keterbatasan sumber daya.

Model kecil memungkinkan:

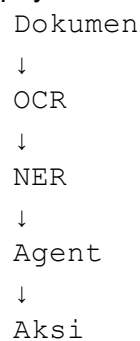
- Berjalan di laptop
- Berjalan di mini server
- Berjalan di laboratorium sederhana

Tanpa ketergantungan infrastruktur mahal.

10.8 Agentic Workflow

Tahap berikutnya setelah NER adalah agentic workflow.

Konsepnya:



Contoh:

Invoice masuk.

Agent:

- Mengecek vendor
- Mengecek nominal
- Mengecek kontrak
- Mengirim notifikasi

secara otomatis.

10.9 Dari Ekstraksi Menjadi Otomasi

Saat ini:

AI Membaca

Masa depan:

AI Membaca

+

AI Memutuskan

+

AI Menjalankan Proses

Ini akan menjadi perubahan besar dalam dunia administrasi dan bisnis.

10.10 Knowledge Graph

Banyak organisasi memiliki jutaan dokumen.

Masalahnya:

Mereka tidak saling terhubung.

NER memungkinkan pembangunan:

Knowledge Graph

Contoh:

Orang

↓

Bekerja di

↓

Instansi

↓

Menandatangani

↓

Kontrak

Hubungan tersebut tidak terlihat jika hanya menyimpan PDF.

10.11 AI untuk Arsip Nasional

Bayangkan:

Jutaan halaman arsip.

NER dapat mengekstrak:

- Tokoh
- Peristiwa
- Lokasi
- Tahun

Kemudian membangun peta sejarah nasional secara otomatis.

Ini bukan lagi teori.

Teknologinya sudah tersedia.

10.12 AI untuk Rumah Sakit

Bayangkan:

Puluhan tahun rekam medis.

NER dapat membantu:

- Mencari pola penyakit
- Mencari tren pengobatan
- Mengidentifikasi risiko

tentu dengan memperhatikan regulasi dan privasi.

10.13 AI untuk Pendidikan

Bayangkan:

Ribuan skripsi.

NER dapat mengekstrak:

- Topik
- Metodologi
- Dataset
- Kata kunci

Kemudian membangun repositori pengetahuan kampus.

10.14 AI untuk Pemerintah Daerah

Bayangkan:

Puluhan ribu surat masuk.

NER dapat membantu:

- Klasifikasi otomatis
- Ekstraksi metadata
- Routing dokumen
- Analitik kebijakan

Semua berjalan lokal.

10.15 Mengapa Open Source Penting

Banyak organisasi fokus pada teknologi.

Namun melupakan kontrol.

Open source memberikan:

Transparansi
Auditabilitas
Kemandirian
Fleksibilitas
Efisiensi Biaya

Prinsipnya sederhana:

Jika data adalah aset strategis, maka kendali atas teknologi juga harus strategis.

10.16 Komunitas Open Source Indonesia

Indonesia memiliki komunitas yang sangat besar.
Bidang yang berkembang cepat:

- Linux
- Python
- Data Science
- AI
- Cyber Security
- DevOps
- Open Hardware

Bergabunglah.
Belajar tidak harus sendirian.

10.17 Komunitas Global

Kontribusi tidak harus besar.

Mulailah dari:

- Dokumentasi
- Bug report
- Tutorial
- Dataset
- Terjemahan

Kontribusi kecil yang dilakukan terus menerus sering menghasilkan dampak besar.

10.18 Ide Proyek Lanjutan

Setelah buku ini selesai, lanjutkan dengan proyek yang lebih besar.
Contoh:

Digitalisasi Arsip Sekolah
Digitalisasi Rekam Medis
Digitalisasi Kontrak
Pencarian Semantik Dokumen
Knowledge Graph Organisasi
Asisten Dokumen Internal
Analitik Surat Masuk
Sistem Audit Otomatis

10.19 Tantangan yang Akan Muncul

Jangan berharap semuanya mudah.

Anda akan menghadapi:

- OCR buruk
- Data kotor
- Dokumen tidak konsisten
- Model gagal
- Infrastruktur bermasalah

Semua itu normal.

Proyek AI selalu lebih banyak tentang data dibanding model.

10.20 Berpikir Seperti Engineer

Jangan bertanya:

Model apa yang terbaik?

Mulailah bertanya:

Masalah apa yang ingin saya selesaikan?

Perbedaan cara berpikir ini sangat penting.

Engineer fokus pada solusi.

Bukan sekadar teknologi.

10.21 Dokumentasikan Semua

Biasakan membuat:

- README
- Diagram
- Log perubahan
- Dokumentasi API

Proyek yang terdokumentasi akan bertahan jauh lebih lama.

10.22 Bagikan Pipeline Anda

Jika berhasil membangun sistem:
Jangan simpan sendiri.

Bagikan:

- Artikel
- Tutorial
- Video
- Presentasi
- Repositori

Semakin banyak orang belajar, semakin kuat ekosistem.

10.23 Bangun Dataset Lokal

Salah satu kontribusi terbesar adalah dataset.

Contoh:

- Arsip Indonesia
- Dokumen pendidikan
- Dokumen hukum
- Dokumen kesehatan yang telah dianonimkan

Dataset berkualitas sering lebih berharga daripada model baru.

10.24 Membangun Kedaulatan AI

Kedaulatan AI bukan berarti menolak teknologi luar.

Kedaulatan AI berarti:

- Memahami teknologi
- Menguasai teknologi
- Memodifikasi teknologi
- Mengembangkan teknologi

Bukan sekadar menjadi pengguna.

10.25 Peran Administrator Jaringan dan Praktisi Siber

Administrator jaringan bukan lagi sekadar menjaga konektivitas.

Perannya berkembang menjadi:

- Penjaga infrastruktur AI
- Penjaga data
- Penjaga keamanan
- Penjaga keberlanjutan sistem

Peran ini akan semakin penting.

10.26 Checklist Setelah Buku Selesai

Pastikan Anda telah mampu:

- Menginstal OCR
- Menggunakan Ollama
- Membuat schema NER
- Menggunakan Pydantic
- Mengelola MariaDB
- Membuat pipeline produksi
- Mengelola cache
- Mengelola queue
- Mengelola deployment
- Mengelola monitoring

Jika seluruh poin tercapai, Anda telah memiliki fondasi kuat untuk membangun berbagai solusi AI dokumen.

10.27 Pesan Penutup

Teknologi selalu berubah.
Model akan berganti.
Framework akan berganti.
Bahasa pemrograman mungkin berganti.
Namun satu hal tidak berubah:

Kemampuan memahami masalah, mengelola data, dan membangun solusi nyata akan selalu bernilai.

OCR dan NER hanyalah pintu masuk menuju dunia yang lebih besar. Dunia di mana dokumen tidak lagi menjadi arsip pasif, melainkan sumber pengetahuan yang hidup dan dapat dimanfaatkan untuk meningkatkan layanan, mempercepat proses, dan membuka wawasan baru.

Bangun sistem yang:

- Transparan
- Dapat diaudit
- Aman
- Bermanfaat

Bangun teknologi yang:

- Memberdayakan
- Mendidik
- Membuka akses pengetahuan

Dan yang paling penting:

Bangun AI yang membantu manusia menjadi lebih baik, bukan AI yang membuat manusia kehilangan kendali atas data, pengetahuan, dan masa depannya.

Selamat Berkarya

Anda kini memiliki seluruh fondasi untuk membangun sistem **OCR + NER + LLM lokal** menggunakan Ubuntu, Python, Ollama, MariaDB, dan perangkat lunak *open source* lainnya. Langkah berikutnya bukan membaca buku lain, tetapi mulai membangun, menguji, memperbaiki, dan membagikan solusi yang Anda buat kepada komunitas.

Karena masa depan AI tidak dibangun oleh mereka yang hanya menggunakan teknologi.

Masa depan AI dibangun oleh mereka yang memahami, memodifikasi, dan berkontribusi terhadap teknologi tersebut.

Lampiran A - Daftar Perintah CLI Lengkap Ubuntu + Ollama

A.1 Update Sistem

```
sudo apt update
sudo apt upgrade -y
sudo apt autoremove -y
```

Cek versi sistem:

```
lsb_release -a
uname -r
hostnamectl
```

A.2 Monitoring Server

```
htop
free -h
df -h
df -i
vmstat 1
iostat -xz 1
```

Instal tool monitoring:

```
sudo apt install htop sysstat ntop curl git unzip -y
```

A.3 GPU

```
lspci | grep -i vga
nvidia-smi
watch -n 1 nvidia-smi
nvidia-smi
```

A.4 Python Environment

```
sudo apt install python3 python3-pip python3-venv python3-dev
build-essential -y
```

```
mkdir ner-ocr-project
cd ner-ocr-project
```

```
python3 -m venv .venv
source .venv/bin/activate
```

```
pip install --upgrade pip
```

A.5 Library Python Utama

```
pip install ollama pydantic pymysql opencv-python pandas  
numpy loguru fastapi uvicorn
```

Untuk OCR:

```
pip install paddleocr paddlepaddle pdf2image pillow
```

A.6 Instalasi Ollama

```
curl -fsSL https://ollama.com/install.sh | sh
```

Ollama menyediakan CLI, API, dan model lokal; dokumentasi resminya juga menekankan penggunaan model terbuka dan eksekusi pada perangkat sendiri.

([Ollama](#))

Cek versi:

```
ollama --version
```

Cek service:

```
systemctl status ollama
```

Aktifkan otomatis saat boot:

```
sudo systemctl enable ollama  
sudo systemctl restart ollama
```

Log:

```
journalctl -u ollama -f
```

A.7 Manajemen Model Ollama

```
ollama list  
ollama pull qwen3  
ollama pull llama3.3  
ollama pull gemma3  
ollama pull phi4  
ollama run qwen3  
ollama show qwen3  
ollama rm qwen3
```

A.8 MariaDB

```
sudo apt install mariadb-server -y
sudo systemctl enable mariadb
sudo systemctl start mariadb
sudo systemctl status mariadb
```

Masuk:

```
sudo mariadb
```

Backup:

```
mariadb-dump -u root -p nerdb > backup_nerdb.sql
```

Dokumentasi resmi MariaDB menempatkan **CREATE TABLE**, **INSERT**, dan utilitas dump sebagai bagian penting untuk penggunaan produksi dan pemulihan data. ([MariaDB](#))

Lampiran B - Template Prompt NER Terbaik + Pydantic Schema

B.1 Prompt NER Umum

Anda adalah sistem ekstraksi entitas dokumen.

Tugas:

Ekstrak entity dari teks OCR berikut.

Entity yang dicari:

- nama
- alamat
- nomor_identitas
- tanggal
- organisasi
- nominal
- produk

Aturan:

- Output harus JSON valid.
- Jangan menambahkan penjelasan.
- Jangan menggunakan markdown.
- Jika data tidak ditemukan, isi null.
- Jangan mengarang data.

Teks OCR:

{text}

B.2 Prompt KTP

Ekstrak data KTP dari teks OCR berikut.

Field:

- nik
- nama
- tempat_lahir
- tanggal_lahir
- alamat
- agama
- pekerjaan

Output JSON valid.

Jika field tidak ditemukan, isi null.

Jangan mengarang.

Teks:

{text}

B.3 Prompt Invoice

Ekstrak data invoice dari teks OCR berikut.

Field:

- vendor
- nomor_invoice
- tanggal
- subtotal
- pajak
- total
- daftar_item

Untuk daftar_item, gunakan array JSON berisi:

- nama_produk
- qty
- harga_satuan
- total_item

Output JSON valid.

Jangan menambahkan komentar.

Teks:

{text}

B.4 Pydantic Schema Umum

```
from pydantic import BaseModel
from typing import Optional, List

class GenericNER(BaseModel):
    nama: Optional[str] = None
    alamat: Optional[str] = None
    nomor_identitas: Optional[str] = None
    tanggal: Optional[str] = None
    organisasi: Optional[str] = None
    nominal: Optional[float] = None
    produk: Optional[str] = None
```

Pydantic dipakai untuk membuat kontrak data, validasi, dan JSON schema agar output LLM lebih mudah dikendalikan. ([Pydantic Docs](#))

B.5 Pydantic Schema KTP

```
from pydantic import BaseModel
from typing import Optional

class KTPSchema(BaseModel):
    nik: Optional[str] = None
    nama: Optional[str] = None
    tempat_lahir: Optional[str] = None
    tanggal_lahir: Optional[str] = None
    alamat: Optional[str] = None
    agama: Optional[str] = None
    pekerjaan: Optional[str] = None
```

B.6 Pydantic Schema Invoice

```
from pydantic import BaseModel
from typing import Optional, List

class InvoiceItem(BaseModel):
    nama_produk: Optional[str] = None
    qty: Optional[int] = None
    harga_satuan: Optional[float] = None
    total_item: Optional[float] = None

class InvoiceSchema(BaseModel):
    vendor: Optional[str] = None
    nomor_invoice: Optional[str] = None
    tanggal: Optional[str] = None
    subtotal: Optional[float] = None
    pajak: Optional[float] = None
    total: Optional[float] = None
    daftar_item: List[InvoiceItem] = []
```

B.7 Structured Output Ollama

Ollama mendukung *structured outputs* berbasis JSON schema sehingga respons model dapat dipaksa mengikuti format tertentu; ini penting untuk ekstraksi dokumen, konsistensi JSON, dan integrasi database. ([Ollama Documentation](#))

Contoh sederhana:

```
from ollama import chat
from pydantic import BaseModel

class KTP(BaseModel):
    nik: str | None = None
    nama: str | None = None
    alamat: str | None = None

response = chat(
    model="qwen3",
    messages=[
        {
            "role": "user",
            "content": "Ekstrak NIK, nama, dan alamat
dari teks OCR berikut..."
        }
    ],
    format=KTP.model_json_schema()
)

print(response.message.content)
```

Lampiran C - MariaDB Schema & Contoh Query

C.1 Membuat Database

```
CREATE DATABASE nerdb
CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
```

C.2 Membuat User

```
CREATE USER 'neruser'@'localhost'
IDENTIFIED BY 'passwordku';

GRANT ALL PRIVILEGES
ON nerdb.*
TO 'neruser'@'localhost';

FLUSH PRIVILEGES;
```

C.3 Tabel Dokumen

```
CREATE TABLE documents (
    id BIGINT AUTO_INCREMENT,
    filename VARCHAR(255) NOT NULL,
    document_type VARCHAR(100),
    file_hash CHAR(64),
    ocr_text LONGTEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY(id),
    UNIQUE(file_hash)
);
```

C.4 Tabel Entity Umum

```
CREATE TABLE entities (
    id BIGINT AUTO_INCREMENT,
    document_id BIGINT,
    entity_type VARCHAR(100),
    entity_value TEXT,
    confidence DECIMAL(5,4),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY(id),
    FOREIGN KEY(document_id) REFERENCES documents(id)
);
```

C.5 Tabel KTP

```
CREATE TABLE ktp_records (  
    id BIGINT AUTO_INCREMENT,  
    document_id BIGINT,  
    nik VARCHAR(32),  
    nama VARCHAR(255),  
    tempat_lahir VARCHAR(255),  
    tanggal_lahir DATE,  
    alamat TEXT,  
    agama VARCHAR(100),  
    pekerjaan VARCHAR(255),  
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    PRIMARY KEY(id),  
    UNIQUE(nik),  
    FOREIGN KEY(document_id) REFERENCES documents(id)  
);
```

C.6 Tabel Invoice

```
CREATE TABLE invoices (  
    id BIGINT AUTO_INCREMENT,  
    document_id BIGINT,  
    vendor VARCHAR(255),  
    nomor_invoice VARCHAR(255),  
    tanggal DATE,  
    subtotal DECIMAL(18,2),  
    pajak DECIMAL(18,2),  
    total DECIMAL(18,2),  
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    PRIMARY KEY(id),  
    FOREIGN KEY(document_id) REFERENCES documents(id)  
);
```

C.7 Tabel Item Invoice

```
CREATE TABLE invoice_items (  
    id BIGINT AUTO_INCREMENT,  
    invoice_id BIGINT,  
    nama_produk VARCHAR(255),  
    qty INT,  
    harga_satuan DECIMAL(18,2),  
    total_item DECIMAL(18,2),  
    PRIMARY KEY(id),  
    FOREIGN KEY(invoice_id) REFERENCES invoices(id)  
);
```

C.8 Insert Dokumen

```
INSERT INTO documents  
(filename, document_type, file_hash, ocr_text)  
VALUES  
( 'invoice001.pdf', 'invoice', 'abc123hash', 'hasil OCR...');
```

C.9 Insert Entity

```
INSERT INTO entities  
(document_id, entity_type, entity_value, confidence)  
VALUES  
(1, 'vendor', 'PT Contoh Indonesia', 0.9500);
```

C.10 Insert Aman dengan Duplikasi

```
INSERT INTO ktp_records  
(document_id, nik, nama, alamat)  
VALUES  
(1, '3173012345678901', 'Budi Santoso', 'Tangerang Selatan')  
ON DUPLICATE KEY UPDATE  
nama = VALUES(nama),  
alamat = VALUES(alamat);
```

MariaDB mendukung operasi **INSERT**, termasuk skenario produksi seperti bulk insert dan penanganan duplikasi. ([MariaDB](#))

Lampiran D - Daftar Model Ollama Rekomendasi 2026

D.1 Prinsip Memilih Model

Pilih model berdasarkan:

- **Konsistensi JSON**
- **Kemampuan multilingual**
- **Kecepatan lokal**
- **Kebutuhan VRAM/RAM**
- **Kemampuan memahami dokumen panjang**
- **Stabilitas untuk *structured output***

D.2 Model Rekomendasi

Model	Kegunaan	Catatan
qwen3	NER multilingual, dokumen Indonesia	Sangat kuat untuk instruksi dan JSON
llama3.3	General NER, reasoning, dokumen panjang	Stabil untuk produksi
llama3.2	Komputer lokal / edge	Lebih ringan
gemma3	Sistem kecil, cepat	Cocok untuk eksperimen
phi4	Reasoning ringan	Cocok untuk lab
mistral	General purpose	Seimbang
deepseek-r1	Reasoning	Perlu validasi ketat untuk JSON
richardyoung/schematron-3b:Q4_K_M	Ekstraksi HTML ke JSON	Fokus schema-valid output

Ollama menyediakan katalog model publik dan dukungan pustaka Python/JavaScript; halaman model resmi juga memuat model komunitas seperti Schematron yang dirancang untuk ekstraksi terstruktur. ([Ollama](#))

D.3 Rekomendasi Praktis

Laptop / CPU Only

```
ollama pull llama3.2  
ollama pull gemma3
```

Server GPU 8–12 GB VRAM

```
ollama pull qwen3  
ollama pull phi4
```

Server GPU Lebih Besar

```
ollama pull llama3.3  
ollama pull qwen3
```

Ekstraksi JSON Ketat

```
ollama pull qwen3
```

Gunakan *temperature* rendah.

Lampiran E - Resources, Dataset Open Source, dan FAQ Troubleshooting

E.1 Resources Open Source

OCR

- PaddleOCR
- Tesseract OCR
- EasyOCR
- DocTR

LLM Lokal

- Ollama
- llama.cpp
- vLLM
- Open WebUI

Python

- Pydantic
- PyMySQL
- OpenCV
- Pandas
- NumPy
- FastAPI
- Loguru

Database

- MariaDB
- SQLite untuk prototipe kecil

E.2 Dataset Open Source

Gunakan dataset legal dan aman.

Contoh jenis dataset:

- Dataset OCR dokumen publik
- Dataset invoice sintetis
- Dataset formulir publik
- Dataset NER multilingual
- Dataset arsip yang sudah bebas hak akses
- Dataset medis yang sudah dianonimkan

Jangan gunakan data pribadi asli tanpa izin.

E.3 FAQ Troubleshooting

1. Ollama tidak jalan

Cek:

```
systemctl status ollama  
journalctl -u ollama -f
```

Restart:

```
sudo systemctl restart ollama
```

2. Model lambat

Solusi:

- Gunakan model lebih kecil
- Kurangi panjang teks OCR
- Aktifkan cache
- Pecah dokumen per halaman

3. Output JSON rusak

Solusi:

- Gunakan *structured output*
- Gunakan Pydantic schema
- Turunkan *temperature*
- Tambahkan instruksi “Output JSON valid saja”

4. Model mengarang data

Solusi:

- Tambahkan instruksi “jangan mengarang”
- Field tidak ditemukan harus `null`
- Validasi dengan regex
- Simpan confidence score

5. OCR buruk

Solusi:

- Deskew
- Denoise
- Binarization
- Contrast enhancement
- Scan ulang minimal 300 DPI

6. MariaDB gagal konek

Cek service:

```
systemctl status mariadb
```

Cek port:

```
ss -tulpn | grep 3306
```

Tes login:

```
mariadb -u neruser -p nerdb
```

7. Database duplikat

Gunakan:

```
UNIQUE(file_hash);
```

atau:

```
ON DUPLICATE KEY UPDATE
```

8. Proses ribuan dokumen lambat

Solusi:

- Gunakan queue
- Gunakan worker paralel
- Gunakan batch insert
- Cache hasil OCR dan NER

9. Memory habis

Solusi:

- Gunakan model lebih kecil
- Kurangi batch size
- Tambah swap
- Pisahkan OCR dan NER worker

10. Hasil NER tidak konsisten

Solusi:

- Pakai schema
- Pakai few-shot prompt
- Gunakan model yang lebih kuat untuk JSON
- Lakukan validasi pasca-ekstraksi

Penutup Lampiran

Lampiran ini melengkapi 10 bab utama buku: dari fondasi konsep, setup Ubuntu, OCR, Ollama, pipeline Python, MariaDB, produksi, hingga masa depan NER lokal. Arah besarnya tetap sama: **bangun AI dokumen yang berjalan lokal, dapat diaudit, hemat biaya, dan tidak mengikat organisasi pada layanan eksternal.**

Glossary

Istilah	Definisi
Accuracy	Persentase hasil prediksi yang benar dibandingkan seluruh data yang diuji.
Agentic Workflow	Alur kerja AI yang mampu mengambil keputusan dan menjalankan aksi secara otomatis berdasarkan hasil analisis.
API (<i>Application Programming Interface</i>)	Antarmuka yang memungkinkan aplikasi saling berkomunikasi secara otomatis.
Audit Trail	Catatan aktivitas sistem yang digunakan untuk pelacakan, investigasi, dan kepatuhan.
Batch Processing	Pemrosesan banyak dokumen atau data secara bersamaan dalam satu kelompok.
Binarization	Teknik mengubah gambar menjadi hitam-putih untuk meningkatkan kualitas OCR.
Bounding Box	Koordinat area teks atau objek yang terdeteksi pada gambar.
Cache	Penyimpanan sementara untuk mempercepat proses dengan menghindari perhitungan ulang.
CER (<i>Character Error Rate</i>)	Metrik untuk mengukur jumlah kesalahan karakter pada hasil OCR.
Chain-of-Thought (CoT)	Teknik prompt yang meminta model berpikir secara bertahap sebelum menghasilkan jawaban.
CLI (<i>Command Line Interface</i>)	Antarmuka berbasis perintah teks untuk berinteraksi dengan sistem operasi atau aplikasi.
Confidence Score	Nilai yang menunjukkan tingkat keyakinan model terhadap hasil prediksi.
Container	Lingkungan terisolasi yang digunakan untuk menjalankan aplikasi secara konsisten di berbagai sistem.
CSV (<i>Comma Separated Values</i>)	Format file teks untuk menyimpan data tabular.
Data Sovereignty	Prinsip bahwa data tetap berada di bawah kendali organisasi atau negara pemilik data.
Database	Sistem penyimpanan data terstruktur yang dapat diakses dan dikelola secara efisien.

Deep Learning	Cabang <i>machine learning</i> yang menggunakan jaringan saraf berlapis banyak untuk mempelajari pola kompleks.
Deduplikasi	Proses mengidentifikasi dan menghapus data yang sama atau berulang.
Deskew	Proses meluruskan dokumen atau gambar yang miring sebelum OCR.
Denoise	Teknik menghilangkan gangguan visual (<i>noise</i>) pada gambar.
Deployment	Proses menempatkan aplikasi ke lingkungan produksi agar dapat digunakan pengguna.
Docker	Platform <i>containerization</i> untuk membangun, mendistribusikan, dan menjalankan aplikasi.
Embedding	Representasi numerik dari teks atau objek yang digunakan untuk pencarian semantik dan analisis AI.
Entity	Informasi penting yang diekstraksi dari teks, seperti nama, lokasi, organisasi, atau nomor identitas.
Entity Linking	Proses menghubungkan entitas yang diekstraksi dengan referensi atau identitas yang konsisten.
ETL (<i>Extract Transform Load</i>)	Proses mengambil, mengubah, dan memuat data ke sistem tujuan.
FastAPI	<i>Framework</i> Python modern untuk membangun API dengan performa tinggi.
Fine-Tuning	Proses menyesuaikan model AI menggunakan dataset tertentu agar lebih sesuai dengan kebutuhan spesifik.
GPU (<i>Graphics Processing Unit</i>)	Perangkat keras yang mempercepat komputasi paralel, termasuk inferensi AI.
Hallucination	Kondisi ketika model AI menghasilkan informasi yang tidak terdapat dalam data sumber.
Hash	Nilai unik yang dihasilkan dari data untuk identifikasi atau verifikasi integritas.
JSON (<i>JavaScript Object Notation</i>)	Format pertukaran data yang ringan dan mudah dibaca manusia maupun mesin.
Knowledge Graph	Struktur data berbasis relasi yang menghubungkan entitas dan hubungan antar entitas.

Latency	Waktu yang dibutuhkan sistem untuk menyelesaikan satu proses atau permintaan.
Linux	Keluarga sistem operasi <i>open source</i> yang banyak digunakan untuk server dan AI.
LLM (<i>Large Language Model</i>)	Model bahasa berukuran besar yang dilatih menggunakan data teks dalam jumlah sangat besar.
LoRA (<i>Low Rank Adaptation</i>)	Teknik <i>fine-tuning</i> ringan yang lebih hemat sumber daya dibanding pelatihan ulang penuh.
Logging	Proses pencatatan aktivitas sistem untuk pemantauan dan analisis.
MariaDB	Sistem manajemen basis data relasional <i>open source</i> .
Metadata	Data yang menjelaskan informasi lain, misalnya tanggal dokumen atau penulis dokumen.
Multilingual	Kemampuan sistem untuk memahami atau menghasilkan banyak bahasa.
Multimodal	Kemampuan model untuk memproses berbagai jenis data seperti teks, gambar, audio, atau video secara bersamaan.
NER (<i>Named Entity Recognition</i>)	Teknik AI untuk mengenali dan mengklasifikasikan entitas penting dalam teks.
Noise	Gangguan visual atau data yang dapat menurunkan kualitas hasil OCR atau analisis.
OCR (<i>Optical Character Recognition</i>)	Teknologi untuk mengubah gambar atau hasil scan menjadi teks yang dapat diproses komputer.
Ollama	Platform untuk menjalankan model AI secara lokal pada perangkat sendiri.
Open Source	Perangkat lunak dengan kode sumber yang dapat dipelajari, dimodifikasi, dan didistribusikan secara bebas.
PaddleOCR	Mesin OCR <i>open source</i> yang mendukung banyak bahasa dan layout dokumen kompleks.
PDF (<i>Portable Document Format</i>)	Format dokumen digital yang umum digunakan untuk distribusi dokumen.
Pydantic	Pustaka Python untuk validasi data dan pembuatan schema berbasis tipe data.

Python	Bahasa pemrograman yang banyak digunakan dalam AI, data science, dan otomasi.
Queue	Mekanisme antrean yang digunakan untuk mengatur urutan pekerjaan atau dokumen yang akan diproses.
Recall	Persentase entitas yang berhasil ditemukan dibandingkan seluruh entitas yang sebenarnya ada.
Regex (<i>Regular Expression</i>)	Pola pencarian teks yang digunakan untuk validasi atau ekstraksi informasi.
REST API	Arsitektur layanan web yang memungkinkan komunikasi antar aplikasi menggunakan protokol HTTP.
Retry Logic	Mekanisme untuk mencoba kembali proses yang gagal secara otomatis.
Schema	Struktur data yang mendefinisikan format, tipe, dan aturan validasi data.
Semantic Search	Teknik pencarian yang memahami makna dan konteks, bukan hanya mencocokkan kata kunci.
Small Language Model (SLM)	Model bahasa berukuran kecil yang lebih ringan dan efisien dibanding LLM besar.
Structured Output	Output AI yang mengikuti format tertentu, misalnya JSON atau schema yang telah ditentukan.
Systemd	Sistem manajemen layanan dan proses pada Linux modern.
Table Extraction	Proses mendeteksi dan mengekstrak data tabel dari dokumen atau gambar.
Throughput	Jumlah dokumen atau transaksi yang dapat diproses dalam satuan waktu tertentu.
Transaction	Sekumpulan operasi database yang dijalankan sebagai satu kesatuan dan dapat dibatalkan jika terjadi kesalahan.
Ubuntu	Distribusi Linux yang banyak digunakan untuk server, AI, dan komputasi ilmiah.
Validation	Proses memastikan data memenuhi aturan dan format yang telah ditentukan.
Vector Database	Basis data yang dirancang untuk menyimpan dan mencari data berbentuk vektor atau embedding.

Vector Search	Teknik pencarian berbasis embedding yang memungkinkan pencarian semantik.
Virtual Environment	Lingkungan Python terisolasi untuk mengelola dependensi proyek secara terpisah.
WebUI	Antarmuka berbasis web untuk mengakses dan mengelola aplikasi.
WER (<i>Word Error Rate</i>)	Metrik untuk mengukur jumlah kesalahan kata pada hasil OCR.
Worker	Proses atau layanan yang bertugas menjalankan pekerjaan tertentu dalam pipeline pemrosesan.
Zero-Shot Learning	Kemampuan model menjalankan tugas baru tanpa pelatihan tambahan khusus pada tugas tersebut.

Referensi

Alammar, J. (2024). *The Illustrated Transformer*. Jalammar.

<https://jalammar.github.io/illustrated-transformer/>

Apache Software Foundation. (2025). *Apache Arrow Documentation*. Apache Software Foundation. <https://arrow.apache.org/docs/>

Canonical Ltd. (2026). *Ubuntu 26.04 LTS Documentation*. Canonical Ltd.

<https://documentation.ubuntu.com/>

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., ... Zaremba, W. (2021). *Evaluating large language models trained on code*. arXiv. <https://arxiv.org/abs/2107.03374>

DeepSeek AI. (2025). *DeepSeek Models Documentation*. DeepSeek AI.

<https://github.com/deepseek-ai>

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT 2019*, 4171–4186. <https://arxiv.org/abs/1810.04805>

DocTR Team. (2025). *DocTR Documentation*. Mindee. <https://mindee.github.io/doctr/>

European Commission. (2024). *eArchiving Building Block Documentation*. Publications Office of the European Union. <https://digital-strategy.ec.europa.eu/>

FastAPI Contributors. (2026). *FastAPI Documentation*. FastAPI Project.

<https://fastapi.tiangolo.com/>

Han, X., Gao, T., Lin, Y., Peng, H., Yang, Y., Xiao, C., Liu, Z., Li, P., Sun, M., & Zhou, J. (2021). More data, more relations, more context and more openness: A review and outlook for relation extraction. *AAACL-IJCNLP 2021*, 745–758. <https://arxiv.org/abs/2004.03186>

Hugging Face. (2026). *Transformers Documentation*. Hugging Face.

<https://huggingface.co/docs/transformers>

LangChain Inc. (2026). *LangChain Documentation*. LangChain.

<https://python.langchain.com/>

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W. T., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33. <https://arxiv.org/abs/2005.11401>

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. arXiv. <https://arxiv.org/abs/1907.11692>

- Loguru Contributors.** (2026). *Loguru Documentation*. Loguru Project. <https://loguru.readthedocs.io/>
- MariaDB Foundation.** (2026). *MariaDB Server Documentation*. MariaDB Foundation. <https://mariadb.com/docs/>
- Meta AI.** (2025). *Llama 3 Model Card and Documentation*. Meta AI. <https://ai.meta.com/>
- Mikolov, T., Chen, K., Corrado, G., & Dean, J.** (2013). Efficient estimation of word representations in vector space. arXiv. <https://arxiv.org/abs/1301.3781>
- Mindee.** (2025). *DocTR: Document Text Recognition Documentation*. Mindee. <https://mindee.github.io/doctr/>
- NVIDIA Corporation.** (2026). *NVIDIA System Management Interface (nvidia-smi) Documentation*. NVIDIA Corporation. <https://docs.nvidia.com/>
- Ollama.** (2026). *Ollama Documentation*. Ollama. <https://ollama.com/>
- Ollama.** (2026). *Structured Outputs Documentation*. Ollama. <https://docs.ollama.com/capabilities/structured-outputs>
- OpenCV Team.** (2026). *OpenCV Documentation*. OpenCV Foundation. <https://docs.opencv.org/>
- PaddlePaddle Authors.** (2026). *PaddleOCR Documentation*. PaddlePaddle. <https://www.paddleocr.ai/>
- Python Software Foundation.** (2026). *Python 3 Documentation*. Python Software Foundation. <https://docs.python.org/3/>
- Pydantic Team.** (2026). *Pydantic Documentation*. Pydantic Project. <https://docs.pydantic.dev/>
- Qwen Team.** (2025). *Qwen Technical Report*. Alibaba Cloud. <https://qwenlm.github.io/>
- Raschka, S.** (2024). *Build a Large Language Model (From Scratch)*. Manning Publications.
- Sentence Transformers Team.** (2026). *Sentence Transformers Documentation*. UKPLab. <https://www.sbert.net/>
- Smith, R.** (2007). An overview of the Tesseract OCR engine. *Proceedings of the Ninth International Conference on Document Analysis and Recognition*, 629–633. IEEE. <https://doi.org/10.1109/ICDAR.2007.4376991>
- Tesseract OCR Project.** (2026). *Tesseract OCR Documentation*. Google Research/Open Source Community. <https://tesseract-ocr.github.io/>
- The Linux Foundation.** (2026). *systemd Documentation*. The Linux Foundation. <https://systemd.io/>

Van Rossum, G., & Drake, F. L. (2025). *Python Language Reference Manual*. Python Software Foundation. <https://docs.python.org/>

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30. <https://arxiv.org/abs/1706.03762>

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T. L., Gugger, S., ... Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. *Proceedings of EMNLP 2020: System Demonstrations*, 38–45. <https://arxiv.org/abs/1910.03771>

W3C. (2025). *JSON Schema Specification*. World Wide Web Consortium. <https://json-schema.org/>

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. *European Conference on Computer Vision*, 818–833. Springer. <https://arxiv.org/abs/1311.2901>

Tentang Penulis



Onno W. Purbo, saat ini bertugas sebagai rektor di Institut Teknologi Tangerang Selatan (ITTS). Onno memperoleh gelar Ph.D bidang Electrical Engineering dari University of Waterloo, Canada, adalah seorang copyleft, educator dan ICT evangelist. Dia sudah mempublikasikan 50++ buku, termasuk free ICT ebook untuk sekolah tahun 2008. Beberapa buku terakhirnya adalah "Internet-TCP/IP: Konsep Dan Implementasi", 2018; "Sistem Operasi, Konsep Dan Membuat Linux OpenWRT Dan ROM Android", 2019; "IPv6 Untuk Mendukung Operasi Jaringan Dan Domain Name System", 2019; "Kubernetes untuk Pemula", 2024 dan "Membuat Operator Seluler 5G Sendiri", 2024.

Dia memimpin sambungan pertama Internet di Institut Teknologi Bandung, tahun 1993-2000, dan menggunakannya untuk membuat jaringan Internet pendidikan yang pertama di Indonesia. Dia membebaskan frekuensi WiFi, memperkenalkan RT/RW-net, antenna Wajanbolic dan jaringan selular OpenBTS dan private 5G sendiri. Dia memimpin jaringan telepon pertama di atas Internet, VoIP Merdeka, yang kemudian hari dikenal sebagai VoIP Rakyat berbasis SIP dan menggunakan kode area +62520 dan +62521. Dia saat ini aktif memperkenalkan e-Learning, dan menjalankan server e-Learning di <http://lms.onnocenter.or.id/moodle/> 66,000++ siswa / mahasiswa dan <https://opencourse.itts.ac.id> 39.000+ mahasiswa secara gratis. Email: onno@indo.net.id