# PENERAPAN OCR DAN NLP UNTUK OTOMASI PENGISIAN ARSIP DIGITAL DAN VALIDASI SURAT PADA APLIKASI ARSIP DIGITAL BERBASIS WEBSITE DI PT NUSA RAYA CIPTA TBK

Fajrul Kamal[1*], Ir. Aghus Sofwan, S.T., M.T., Ph.D., IPU.[2] dan Yuli Christyono, S.T., M.T.[3]

[123]Departemen Teknik Elektro, Fakultas Teknik, Universitas Diponegoro, Semarang, Indonesia

[*]*Penulis korespondensi, E-mail: fajrulkamal@students.undip.ac.id*

## Abstrak

*PT Nusa Raya Cipta Tbk masih menggunakan sistem pengarsipan dokumen secara manual, yang menghambat efisiensi serta aksesibilitas dalam pengelolaan administrasi perusahaan. Untuk mengatasi permasalahan tersebut, penelitian ini bertujuan untuk mengembangkan sistem arsip digital guna meningkatkan efisiensi serta kemudahan akses terhadap dokumen. Sistem yang dikembangkan berbasis Machine Learning dan mengintegrasikan Optical Character Recognition (OCR) untuk mengekstraksi teks dari dokumen fisik, serta Natural Language Processing (NLP) untuk menerjemahkan teks dari Bahasa Indonesia ke Bahasa Inggris secara otomatis. Implementasi OCR dilakukan dengan menyesuaikan konfigurasi serta memanfaatkan regex untuk mendeteksi pola teks tertentu, guna memastikan ekstraksi data yang lebih akurat dan sesuai dengan kebutuhan sistem. Pengembangan sistem menggunakan metode Kanban secara iteratif, memungkinkan fleksibilitas tinggi dalam menyesuaikan perubahan serta penyempurnaan fitur secara berkelanjutan. Proses pengembangan mencakup integrasi OCR dengan regex untuk otomatisasi pengarsipan dokumen dan pemrosesan NLP untuk penerjemahan bahasa. Hasil penelitian ini menunjukkan bahwa sistem arsip digital yang dikembangkan mampu mengubah dokumen fisik menjadi format digital yang dapat diakses, diindeks, dan dianalisis secara optimal. Sistem ini meningkatkan efisiensi pencarian serta pengelolaan dokumen, sehingga mendukung peningkatan produktivitas dan efektivitas dalam administrasi perusahaan.*

*Kata kunci: Digitalisasi Arsip, Optical Character Recognition (OCR), Natural Language Processing (NLP), Machine learning, Penerjemahan Otomatis, Regex, Pengelolaan Dokumen, Kanban*

## Abstract

*PT Nusa Raya Cipta Tbk still employs a manual document archiving system, which hampers efficiency and accessibility in administrative management. To address this issue, this study aims to develop a digital archival system to enhance efficiency and ease of access to documents. The developed system is based on Machine Learning and integrates Optical Character Recognition (OCR) to extract text from physical documents, as well as Natural Language Processing (NLP) to automatically translate text from Indonesian to English. The OCR implementation is configured and utilizes regex to detect specific text patterns, ensuring more accurate data extraction tailored to system requirements. The system development follows an iterative Kanban approach, allowing high flexibility in adapting to changes and refining features continuously. The development process includes the integration of OCR with regex for document archiving automation and NLP processing for language translation. The results of this study indicate that the developed digital archival system can transform physical documents into digital formats that can be accessed, indexed, and analyzed optimally. This system improves the efficiency of document retrieval and management, thereby supporting increased productivity and effectiveness in corporate administration.*

*Keywords: Digital Archiving, Optical Character Recognition (OCR), Natural Language Processing (NLP), Machine learning, Automated Translation, Regex, Document Management, Kanban*

# 1. Pendahuluan

Digitalisasi adalah proses transformasi bisnis dan operasional ke dalam format digital, yang mencakup konversi data menjadi bentuk yang lebih efisien dan terintegrasi [1]. Sejalan dengan perkembangan ini, PT Nusa Raya Cipta Tbk. berencana mengembangkan sistem pencatatan arsip dan persuratan secara digital untuk meningkatkan efisiensi manajemen dokumen. Saat ini, manajemen arsip masih dilakukan secara manual, khususnya di Departemen Administrasi & Sekretariat, sehingga memerlukan lebih banyak sumber daya. Selain itu, penyimpanan dokumen secara digital dapat mengurangi risiko degradasi data akibat kerusakan fisik atau perangkat keras. Teknologi *cloud computing* memungkinkan pengguna mengakses dokumen terbaru dan mengurangi risiko kehilangan data [2].

Pengembangan sistem ini bertujuan untuk mempermudah pengelolaan dokumen penting, seperti dokumen persyaratan tender dan proyek. Implementasi sistem manajemen arsip digital berbasis web terbukti meningkatkan efisiensi operasional, keamanan data, serta memudahkan pendataan dan pencarian arsip [3]. Untuk mendukung digitalisasi, PT NRC mempertimbangkan pemanfaatan teknologi *machine learning*, termasuk *Optical Character Recognition* (OCR) dan *Natural Language Processing* (NLP). OCR memungkinkan konversi dokumen fisik ke teks digital yang dapat diindeks, sehingga mempermudah pencarian informasi dan otomatisasi proses digitalisasi [4].

NLP dapat digunakan untuk menerjemahkan dokumen antara Bahasa Indonesia dan Bahasa Inggris guna mendukung kerja sama PT NRC dengan perusahaan internasional. Dengan NLP, penerjemahan dapat dilakukan secara otomatis dengan tingkat akurasi tinggi [5]. Selain itu, integrasi OpenCV dengan OCR dan regex memberikan manfaat signifikan dalam pemrosesan dokumen digital, seperti meningkatkan akurasi pengenalan teks serta mengekstrak informasi berdasarkan pola tertentu, seperti tanggal atau nomor kontrak [6].

Dalam upaya mencapai tujuan meningkatkan efisiensi perusahaan, integrasi penyimpanan arsip yang terpusat menjadi langkah strategis yang sangat penting. Implementasi sistem digital yang terstruktur tidak hanya memberikan kemudahan dalam pengelolaan dokumen, tetapi juga mendukung efisiensi operasional secara keseluruhan. Dengan penerapan teknologi canggih seperti *artificial intelligence* (AI), *cloud computing*, dan *text recognition*, organisasi dapat mengotomatisasi proses yang sebelumnya dilakukan secara manual, mengurangi kesalahan manusia, dan mempercepat akses serta pemrosesan informasi. Secara keseluruhan, kombinasi teknologi-teknologi ini membentuk fondasi sistem pengelolaan arsip digital yang modern, efektif, dan efisien. Organisasi tidak hanya dapat menghemat sumber daya, tetapi juga menciptakan ekosistem kerja yang lebih produktif dan responsif terhadap tantangan zaman. Integrasi ini mencerminkan evolusi menuju digitalisasi total yang mendukung keberlanjutan dan daya saing perusahaan dalam era transformasi digital [7].

# 2. Metode

## 2.1 Perencanaan Pengembangan Sistem

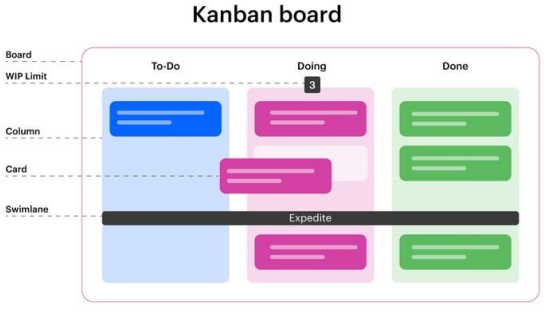
Perencanaan pengembangan sistem dimulai dengan studi literatur, wawancara dengan calon pengguna, dan penentuan alur pengembangan bersama pengembang. Studi literatur mencakup referensi dari jurnal, artikel, ebook, serta analisis sistem serupa yang membahas perancangan modul *machine learning* dalam aplikasi arsip digital. Tujuannya adalah memahami permasalahan yang akan diselesaikan dan mengevaluasi solusi yang telah ada sebagai dasar perancangan sistem.

Wawancara dilakukan secara tidak terstruktur untuk mengidentifikasi jenis dokumen yang dapat diproses menggunakan *machine learning*. Salah satu fokusnya adalah menentukan ada tidaknya pola tetap yang bisa digunakan untuk ekstraksi informasi dengan meminta contoh dokumen aman sebagai referensi. Selain itu, wawancara juga membantu menetapkan dokumen yang memerlukan fitur alih bahasa dari Bahasa Indonesia ke Bahasa Inggris menggunakan NLP.

Pengembangan sistem dilakukan oleh dua pengembang secara paralel dengan pendekatan kanban, yang memudahkan kolaborasi dalam tim kecil. Pendekatan ini memungkinkan pembagian tugas yang fleksibel dan efisien, sehingga setiap bagian sistem dapat dikembangkan secara bersamaan tanpa menghambat alur kerja.

## 2.2 Alur Perancangan Sistem

Dalam perancangan aplikasi arsip digital, metodologi yang digunakan adalah metodologi kanban. Metodologi pengembangan kanban dipilih karena memiliki tahapan yang fleksibel dan dapat meningkatkan efisiensi kerja. Hal ini cocok untuk pengembangan aplikasi arsip digital yang responsif dengan perubahan kebutuhan oleh pengguna. Dengan penggunaan metodologi pengembangan kanban, perubahan kebutuhan fungsional aplikasi oleh pengguna dapat segera ditetapkan dan dikerjakan oleh tim pengembang [8].




Gambar 1 Contoh papan kanban.

## 2.3 *Tools* Perancangan

### 2.3.1. Visual Studio Code

Visual Studio Code (VS Code) adalah editor kode sumber yang dikembangkan oleh Microsoft, dirilis pada tahun 2015. Visual Studio Code merupakan editor kode yang memiliki keunggulan dalam kemudahan penggunaan, dan dapat meningkatkan efisiensi pengguna dalam mengembangkan proyek di dalamnya. Dengan fitur yang intuitif, pengguna dapat dengan cepat mempelajari dan memanfaatkannya untuk berbagai keperluan pemrograman. [9].

Gambar 2 Logo Visual Studio Code

### 2.3.2. Github

GitHub adalah platform *hosting* yang sangat penting dalam pengembangan perangkat lunak, berfungsi sebagai repositori untuk proyek *open-source* dan kolaborasi antar pengembang. GitHub menggunakan sistem kontrol versi Git, yang memungkinkan pengembang untuk menyimpan dan melacak perubahan dalam kode sumber. Sebagai platform berbasis web, GitHub memfasilitasi kolaborasi dengan menyediakan fitur-fitur seperti *pull requests*, *issues*, dan *branch*, yang mendukung kerja tim dalam proyek perangkat lunak [10].


Gambar 3 Logo Github

### 2.3.3. Postman

Postman adalah alat yang populer digunakan oleh pengembang dan penguji perangkat lunak untuk mengembangkan, menguji, dan mendokumentasikan API. Awalnya, Postman merupakan *plug in* sederhana untuk browser Chrome, namun kini telah berkembang menjadi solusi lengkap untuk pengujian API yang digunakan oleh jutaan pengembang dan ribuan perusahaan di seluruh dunia [11]


Gambar 4 Logo Postman

# 3. Hasil dan Pembahasan

## 3.1 Sistem OCR

Modul OCR akan dikembangkan menggunakan bahasa pemrograman Python dengan memanfaatkan pustaka Tesseract OCR dalam melakukan proses ekstraksi teks dari gambar. Sebelum dilakukan pemrosesan ekstraksi teks oleh Tesseract OCR, terlebih dahulu gambar dilakukan pra-pemrosesan menggunakan pustaka OpenCV untuk mengolah dan menyiapkan dokumen agar hasil ekstraksi oleh OCR menjadi lebih akurat.

Teks yang telah dilakukan proses ekstraksi oleh Tesseract OCR kemudian diproses menggunakan pustaka regex untuk memfilter informasi-informasi tertentu saja dari keseluruhan teks yang ada. Hasil ekstraksi oleh regex kemudian dilakukan proses penyesuaian format agar sesuai dengan format yang dibutuhkan oleh aplikasi arsip digital. Keseluruhan informasi hasil ekstraksi oleh regex yang telah dilakukan penyesuaian format akan dikirimkan ke aplikasi arsip digital dalam bentuk JSON.

### 3.1.1. Pra-pemrosesan Dokumen

Pra-pemrosesan merupakan langkah krusial dalam sistem ekstraksi dokumen berbasis OCR. Melalui evaluasi dan optimasi berulang, dirancang metode pra-pemrosesan optimal menggunakan pdf2image, OpenCV, NumPy, dan Pillow untuk meningkatkan akurasi ekstraksi isi dokumen..

A. Konversi PDF ke Gambar

Konversi PDF ke format gambar dilakukan menggunakan dukungan pustaka pdf2image. Berikut contoh kutipan kode proses konversi yang dilakukan.

```
from pdf2image import convert_from_path

def extract_text_from_pdf(pdf_path, doc_type=None, dpi=300):
    # Konversi PDF ke gambar
    images = convert_from_path(pdf_path, dpi=dpi)
```

Gambar 5 Kutipan kode konversi PDF ke gambar

B. Penghapusan Latar Belakang

Penghapusan latar belakang dilakukan dengan cv2.cvtColor dan *K-Means Clustering* dari OpenCV. Gambar dikonversi ke LAB *color space* untuk memisahkan warna latar belakang dan teks. *K-Means* (k=2) mengelompokkan piksel, memungkinkan pemisahan otomatis. Setelah itu, latar belakang dihapus dan hasilnya dikonversi ke *grayscale* untuk meningkatkan keterbacaan sebelum OCR dilakukan..

```python
import cv2
import numpy as np

def remove_background(image, k=2):
    # Konversi gambar ke LAB color space
    lab = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)

    # Reshape gambar untuk K-Means Clustering
    pixel_values = image.reshape((-1, 3)).astype(np.float32)

    # Terapkan K-Means Clustering
    _, labels, centers = cv2.kmeans(pixel_values, k, None,
     (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2), 10, cv2.KMEANS_RANDOM_CENTERS)

    # Konversi kembali hasil clustering ke grayscale
    segmented_image = centers[labels.flatten()].reshape(image.shape)
    return cv2.cvtColor(segmented_image, cv2.COLOR_BGR2GRAY)
```

Gambar 6 Kutipan kode penghapusan latar belakang

C. Peningkatan Kontras

Peningkatan kontras dilakukan dengan fungsi createCLAHE dari OpenCV, yang merupakan metode *Contrast Limited Adaptive Histogram Equalization* (CLAHE). Fungsi ini meningkatkan kontras gambar dengan menyesuaikan distribusi intensitas cahaya pada area kecil secara adaptif.

```python
import cv2

def enhance_contrast(image):
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    return clahe.apply(image)
```

Gambar 7 Kutipan kode peningkatan kontras

D. Pengurangan *Noise*

proses pengurangan noise (denoising) dilakukan dengan menggunakan fungsi fastNlMeansDenoising dari OpenCV. Fungsi ini mengambil parameter utama berupa gambar yang akan diproses, serta nilai parameter h, templateWindowSize, dan searchWindowSize untuk menentukan seberapa besar area yang digunakan dalam perhitungan penghapusan *noise*.

```python
import cv2
import numpy as np

def fast_denoise(image, downscale_factor=2):
    small_image = cv2.resize(image, (image.shape[1] // downscale_factor, image.shape[0] // downscale_factor))
    denoised_small = cv2.fastNlMeansDenoising(small_image, None, h=5, templateWindowSize=5, searchWindowSize=15)
    return cv2.resize(denoised_small, (image.shape[1], image.shape[0]))
```

Gambar 8 Kutipan kode pengurangan *noise*

E. Penerapan *Adaptive Tresholding*

Thresholding mengubah gambar grayscale menjadi hitam-putih (binarisasi) untuk memperjelas teks dengan memisahkan area terang dan gelap. Adaptive thresholding diterapkan dengan cv2.adaptiveThreshold(), yang menerima gambar grayscale, nilai maksimum intensitas (255), metode thresholding (GAUSSIAN_C atau MEAN_C), dan ukuran blok pixel untuk perhitungan threshold.

```python
import cv2

def apply_adaptive_thresholding(image, method="gaussian", block_size=11, C=2):
    if len(image.shape) == 3:
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    if method == "mean":
        result = cv2.adaptiveThreshold(
            image, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, block_size, C
        )
    elif method == "gaussian":
        result = cv2.adaptiveThreshold(
            image, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, block_size, C
        )
    return result
```

Gambar 9 Kutipan kode penerapan *adaptive tresholding*

F. *Deksewing*.

proses *deskewing* (meluruskan teks yang miring) dilakukan dengan menggunakan fungsi minAreaRect dari OpenCV yang menghitung sudut kemiringan teks dalam sebuah gambar berdasarkan kontur objek terang (teks) pada latar belakang gelap. Setelah sudut kemiringan dihitung, gambar kemudian diputar menggunakan getRotationMatrix2D dan warpAffine, sehingga teks yang sebelumnya miring menjadi sejajar dengan horizontal.

```python
import cv2
import numpy as np

def deskew_image(image, angle_threshold=1.5):
    coords = np.column_stack(np.where(image > 0))
    angle = cv2.minAreaRect(coords)[-1]

    if angle < -45:
        angle = -(90 + angle)
    else:
        angle = -angle

    if abs(angle) <= angle_threshold:
        return image  # Tidak perlu deskew jika sudah cukup lurus

    (h, w) = image.shape[:2]
    center = (w // 2, h // 2)
    M = cv2.getRotationMatrix2D(center, angle, 1.0)
    rotated = cv2.warpAffine(image, M, (w, h), flags=cv2.INTER_CUBIC, borderMode=cv2.BORDER_REPLICATE)
    return rotated
```

Gambar 10 Kutipan kode *deskewing*

G. *Upscaling*

*Upscaling* digunakan untuk meningkatkan ukuran gambar tanpa kehilangan terlalu banyak detail. Proses peningkatan resolusi gambar dilakukan menggunakan fungsi *resize* dari OpenCV dengan metode interpolasi bicubic (cv2.INTER_CUBIC).

```python
import cv2
import time

def upscale_image(image, scale_factor=2):
    start_time = time.time()  # Start time tracking

    width = int(image.shape[1] * scale_factor)
    height = int(image.shape[0] * scale_factor)
    dim = (width, height)

    upscaled = cv2.resize(image, dim, interpolation=cv2.INTER_CUBIC)

    end_time = time.time()  # End time tracking
    print(f"Upscaling time: {end_time - start_time:.4f} seconds")

    return upscaled
```

Gambar 11 Kutipan kode *upscaling*

H. Pemotongan Kop Surat

Penghapusan kop surat dilakukan dengan cv2.cvtColor, cv2.threshold, dan cv2.findContours dari OpenCV. Gambar dikonversi ke *grayscale* jika berwarna untuk pemrosesan optimal. *Thresholding* diterapkan untuk mengubah teks menjadi putih dan latar belakang hitam, memudahkan deteksi kontur. cv2.findContours digunakan untuk menemukan kontur, lalu bagian atas gambar ($y < 100$, $w > 500$) diidentifikasi sebagai kop surat. Jika terdeteksi, area tersebut dipotong, menyisakan hanya isi dokumen.

```python
import cv2
import numpy as np

def crop_letterhead(image):
    height, width = image.shape[:2]
    safe_margin = int(height * 0.05)

    # Konversi ke grayscale jika masih berwarna
    if len(image.shape) == 3:
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Terapkan thresholding untuk mendapatkan teks sebagai area putih
    _, thresh = cv2.threshold(image, 200, 255, cv2.THRESH_BINARY_INV)

    # Deteksi kontur teks dalam dokumen
    contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    max_y = 0
    for contour in contours:
        x, y, w, h = cv2.boundingRect(contour)
        if y < 100 and w > 500:  # Cek apakah area ini bagian dari kop surat
            max_y = max(max_y, y + h)

    # Potong bagian kop surat dan simpan bagian isi dokumen
    return image[max_y + safe_margin:, :]
```

Gambar 12 Kutipan kode pemotongan kop surat

### 3.1.2. Ekstraksi OCR

Ekstraksi teks dilakukan menggunakan pytesseract, *wrapper* Python untuk Tesseract OCR. Fungsi image_to_string digunakan untuk mengekstrak teks dari gambar yang telah melalui pra-pemrosesan. Setelah itu, gambar dikonversi ke format Pillow Image (PIL) dengan Image.fromarray(image) agar dapat diproses oleh Tesseract OCR. Proses ekstraksi teks dilakukan dengan image_to_string(), yang secara otomatis mendeteksi dan mengubah karakter dalam gambar menjadi teks.

```python
def ocr_extract(image):
    """Menjalankan OCR pada satu gambar grayscale."""
    return image_to_string(Image.fromarray(image))

def extract_text_from_pdf(pdf_path, doc_type=None, dpi=300):
    """Ekstrak teks dari file PDF menggunakan multi-threading."""
    # Konversi PDF ke gambar
    images = convert_from_path(pdf_path, dpi=dpi)

    # Pra-pemrosesan gambar sebelum OCR
    processed_images = [preprocess_image(img, doc_type) for img in images]

    # Gunakan multi-threading untuk mempercepat ekstraksi OCR
    with concurrent.futures.ThreadPoolExecutor() as executor:
        extracted_texts = list(executor.map(ocr_extract, processed_images))

    # Gabungkan teks dari semua halaman
    text = "\n\n".join(extracted_texts).strip()

    return text
```

Gambar 13 Kutipan kode ekstraksi Tesseract OCR.

### 3.1.3. Ekstraksi Regex

Berdasarkan analisis informasi penting, terdapat 9 jenis dokumen yang diekstrak menggunakan regex: surat keluar, surat masuk, CV, legalitas, tenaga ahli, keuangan, pengurus, pemegang saham, dan kontrak. Penentuan ini mempertimbangkan ketersediaan contoh dokumen dari PT NRC, kemampuan regex dalam mengenali pola, serta keseragaman pola ekstraksi tiap dokumen. Berikut adalah contoh kode pola yang digunakan dalam sistem ekstraksi dokumen berbasis OCR.

```python
# Pola Jenis Dokumen Kontrak
patterns = {
        "tanggal": r"Tanggal\s*:\s*(\d{1,2})\s(\w+)\s(\d{4})",
        "nomor_kontrak": r"No\.\sKontrak\s*:\s*([A-Za-z0-9\-]+)",
        "nama_proyek": r"Nama\sProyek\s*:\s*(.*)",
        "pemberi_kerja": r"Pemberi\sKerja\s*:\s*(.*)"
    }
```

Gambar 14 Kutipan kode contoh pola regex jenis dokumen kontrak.

### 3.1.4. Pengolahan Ekstraksi Regex

Hasil ekstraksi data dari regex diproses dengan re.search(patterns["pattern"], text) untuk mencari teks yang sesuai dengan pola regex. Jika tidak ditemukan kecocokan, nilai dikembalikan sebagai "N/A". Hasil ekstraksi disimpan dalam array hasil dan dikirim dalam format JSON ke aplikasi arsip digital. Untuk memastikan data seragam, dilakukan empat metode pasca-pemrosesan, yaitu pemrosesan tanggal, ekstraksi pola ganda, perhitungan masa berlaku, dan pemrosesan alamat multi-baris.

A. Ekstraksi Tanggal

Isian data tanggal dalam tiap dokumen memiliki format yang bervariasi, seperti DD-MM-YYYY atau format Amerika Serikat (Bulan DD, YYYY), sehingga diperlukan penyeragaman dalam tahap pascapemrosesan. Untuk mengatasi perbedaan ini, dikembangkan fungsi month_mapping dan parse_date agar seluruh tanggal dikonversi ke dalam format "%d-%m-%Y" secara seragam.

```python
def map_month_name():
    month_names = [
        ["Januari", "Jan", "January"], ["Februari", "Feb", "February"],
        ["Maret", "Mar", "March"], ["April", "Apr"], ["Mei", "May"],
        ["Juni", "Jun", "June"], ["Juli", "Jul", "July"],
        ["Agustus", "Agu", "Agt", "Aug", "August"], ["September", "Sep"],
        ["Oktober", "Okt", "Oct", "October"], ["November", "Nov"],
        ["Desember", "Des", "Dec", "December"]
    ]
    return {name.lower(): f"{i+1:02}" for i, names in enumerate(month_names) for name in names}

month_mapping = map_month_name()

def parse_date(date_str, format_hint=None):
    try:
        if format_hint:
            return datetime.strptime(date_str, format_hint)

        # Match full date: "12 Januari 2023"
        match = re.match(r"(\d{1,2})\s+(\w+)\s+(\d{4})", date_str)
        if match:
            day, month_name, year = match.groups()
            month_number = month_mapping.get(month_name.lower())
            if not month_number:
                raise ValueError(f"Bulan tidak dikenali: {month_name}")
            return datetime.strptime(f"{day.zfill(2)}-{month_number}-{year}", "%d-%m-%Y")

        # Match Month Day, Year: "December 31, 2020"
        match = re.match(r"(\w+)\s+(\d{1,2}),\s+(\d{4})", date_str)
        if match:
            month_name, day, year = match.groups()
            month_number = month_mapping.get(month_name.lower())
            if not month_number:
                raise ValueError(f"Bulan tidak dikenali: {month_name}")
            return datetime.strptime(f"{day.zfill(2)}-{month_number}-{year}", "%d-%m-%Y")

        # Match partial date: "Jan 2004" (default day = 1)
        match = re.match(r"(\w+)\s+(\d{4})", date_str)
        if match:
            month_name, year = match.groups()
            month_number = month_mapping.get(month_name.lower())
            if not month_number:
                raise ValueError(f"Bulan tidak dikenali: {month_name}")
            return datetime.strptime(f"01-{month_number}-{year}", "%d-%m-%Y")

        raise ValueError(f"Tanggal tidak valid: {date_str}")
    except Exception as e:
        raise ValueError(f"Error parsing date: {date_str} ({str(e)})")
```

Gambar 15 Kutipan kode pemrosesan tanggal

B. Ekstraksi Pola Ganda

Beberapa dokumen memiliki berbagai format detail informasi dokumen, sehingga regex perlu menangkap lebih dari satu kemungkinan pola. Misalnya, dalam legalitas.py, keuangan.py, dan kontrak.py, regex menangkap nomor dokumen dalam beberapa format, seperti "Nomor: 12345", "PB-UMKU: ABC-678", atau "No. Kontrak: XYZ-999". Untuk mengolah hasil ekstraksi dengan pola ganda diberlakukan kode berikut ini.

```
nomor_dokumen_match = re.search(patterns["nomor_dokumen"], text)
if nomor_dokumen_match:
    hasil["nomor_dokumen"] = nomor_dokumen_match.group(1) or nomor_dokumen_match.group(2)
else:
    hasil["nomor_dokumen"] = "N/A"
```

Gambar 16 Kutipan kode pemrosesan pola ganda

C. Perhitungan Masa Berlaku Dokumen

Pada beberapa dokumen seperti sertifikat tenaga ahli, masa berlaku dapat berupa tanggal eksplisit atau jumlah tahun dari tanggal terbit. Perbedaan jenis penulisan masa berlaku dokumen memerlukan pengolahan lebih lanjut. Berikut merupakan kutipan kode yang digunakan untuk menghitung masa berlaku dokumen.

```
validity_match = re.search(patterns["validity_date"], text)
if validity_match:
    day, month_name, year = validity_match.groups()
    date_str = f"{day} {month_name} {year}"
    hasil["validity"] = parse_date(date_str).strftime("%d-%m-%Y")
else:
    validity_years_match = re.search(patterns["validity_years"], text)
    if validity_years_match and "terbit_date" in hasil and hasil["terbit_date"] != "N/A":
        years = int(validity_years_match.group(1))
        validity_date = parse_date(hasil["terbit_date"]) + relativedelta(years=years)
        hasil["validity"] = validity_date.strftime("%d-%m-%Y")
    else:
        hasil["validity"] = "N/A"
```

Gambar 17 Kutipan kode perhitungan masa berlaku

D. Pemrosesan Alamat Multi Baris

Dalam dokumen tertentu seperti pengurus, pemegang saham, dan CV, alamat sering ditulis dalam beberapa baris. Oleh karena itu, perlu adanya pengolahan kembali agar informasi yang terambil oleh pola regex yang digunakan menjadi seragam sesuai dengan format yang ditentukan. Berikut kutipan kode yang digunakan untuk mengolah isian dengan baris banyak.

```
alamat_match = re.search(patterns["alamat"], text, re.DOTALL)
if alamat_match:
    alamat_lines = alamat_match.group(1).split("\n")
    alamat = ", ".join([line.strip() for line in alamat_lines if line.strip()])
else:
    alamat = "N/A"
```

Gambar 18 Kutipan kode pemrosesan alamat multi baris

### 3.1.5. Pengiriman Hasil Ekstraksi

Konfigurasi *endpoint* dilakukan untuk mengintegrasikan sistem ekstraksi dokumen berbasis OCR dengan aplikasi arsip digital. Tidak seperti model *machine learning* yang dapat langsung di-*deploy*, sistem ini memerlukan *endpoint* karena harus melewati beberapa tahap sebelum dan sesudah ekstraksi menggunakan Tesseract OCR. Pendekatan berbasis *endpoint* juga memungkinkan penyesuaian teknik pra-pemrosesan sesuai kebutuhan tiap dokumen. Berikut adalah kutipan kode konfigurasi *endpoint* menggunakan Flask

```
app = Flask(__name__)
UPLOAD_FOLDER = 'uploads'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['ALLOWED_EXTENSIONS'] = {'pdf'}

def allowed_file(filename):
    """Check if a file has an allowed extension."""
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in app.config['ALLOWED_EXTENSIONS']

@app.route('/extract', methods=['POST'])
def extract_document():
    doc_type = request.form.get('doc_type', 'unknown')
    file = request.files.get('file')

    if not file or not allowed_file(file.filename):
        return jsonify({"error": "Invalid or missing file"}), 400

    filename = secure_filename(file.filename)
    file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    file.save(file_path)

    try:
        # Extract text using the Hybrid Optimized OCR Endpoint
        extracted_text = extract_text_from_pdf(file_path, doc_type=doc_type)

        # Dictionary mapping document types to their respective extractors
        extractors = {
            "legalitas": extract_legalitas,
            "tenaga_ahli": extract_tenaga_ahli,
            "kontrak": extract_kontrak,
            "cv": extract_cv,
            "keuangan": extract_keuangan,
            "surat_masuk": extract_surat_masuk,
            "surat_keluar": extract_surat_keluar,
            "pengurus": extract_pengurus_pemegang_saham,
            "pemegang_saham": extract_pengurus_pemegang_saham
        }

        #  Use the appropriate extractor for the document type
        if doc_type in extractors:
            result = extractors[doc_type](extracted_text)
        else:
            result = {"error": f"Unknown document type: {doc_type}"}

    except Exception as e:
        result = {"error": f"An error occurred: {str(e)}"}

    finally:
        #  Remove the uploaded file after processing
        if os.path.exists(file_path):
            os.remove(file_path)

    return jsonify(result)

if __name__ == '__main__':
    app.run(debug=True)
```

Gambar 19 Kutipan kode pengiriman hasil ekstraksi dengan Flask

## 3.2 Sistem NLP

NLP dibuat untuk menerjemahkan dokumen PDF dari Bahasa Indonesia ke Bahasa Inggris. Modul aplikasi NLP dikembangkan dengan menerapkan pustaka pymupdf untuk pemrosesan dan manipulasi dokumen dengan format PDF yang telah diunggah. Proses penerjemahan teks dilakukan dengan memanfaatkan pustaka deep-translator. Proses pengembangan dimulai dengan menentukan jenis dokumen yang akan dilakukan proses penerjemahan dari Bahasa Indonesia ke Bahasa Inggris. Tahapan selanjutnya adalah melakukan pemrosesan dokumen dari bentuk PDF menjadi teks dan melakukan ekstraksi posisi blok tiap teks pada dokumen. Teks ekstraksi kemudian diterjemahkan berdasarkan potongan posisi blok pada tahapan sebelumnya. Hasil terjemahan kemudian diposisikan ulang seperti pada dokumen aslinya. Setelah itu, dilakukan proses evaluasi berdasarkan kesesuaian hasil terjemahan dengan dokumen asli. Apabila hasilnya tidak sesuai keinginan, dilakukan proses konfigurasi ulang dan percobaan dengan pustaka lainnya apabila diperlukan. Setelah hasil terjemahan dokumen sudah mencapai tingkat keakuratan yang diinginkan dilanjutkan dengan proses konfigurasi *endpoint* untuk keperluan integrasi dengan aplikasi arsip digital untuk mengirimkan dokumen hasil terjemahan ke aplikasi.

### 3.2.1. Konfigurasi Pustaka

Pengembangan sistem pada tahapan ini dilakukan dengan mencari referensi-referensi sistem yang mirip dan melakukan penerapan pada sistem penerjemah otomatis yang dikembangkan. Pada sistem ini, konfigurasi pustaka yang digunakan adalah pustaka pymupdf dan deep_translator. Kedua pustaka ini dipilih karena performanya dalam melakukan operasi penerjemahan terstruktur cukup tinggi. Selain itu, terdapat fungsi yang memungkinkan penyusunan kembali hasil terjemahan menjadi dokumen dengan struktur seperti semula. Hal ini sangat penting karena hasil akhir penerjemahan dokumen harus berupa dokumen PDF bukan berupa teks seperti sistem ekstraksi yang dikembangkan sebelumnya. Pustaka lain yang digunakan adalah Flask yang menyediakan dukungan pemrosesan layanan HTTP untuk integrasi sistem dengan aplikasi arsip digital.

A. Flask

Pustaka Flask digunakan untuk mempermudah proses integrasi dengan aplikasi arsip digital. Flask menawarkan inti yang ringan tanpa lapisan abstraksi atau dependensi berlebihan, memungkinkan pengembang untuk menambahkan komponen sesuai kebutuhan proyek mereka. Fleksibilitas ini menjadikan Flask pilihan populer untuk membangun prototipe API dan aplikasi web yang efisien [12]. Berikut kutipan kode yang menunjukkan penggunaan Flask pada sistem penerjemah otomatis berbasis NLP.

```
from flask import Flask, request, render_template, send_file, jsonify

# Initialize Flask app
app = Flask(__name__)

# Route to display the HTML form for file upload
@app.route('/')
def index():
    return render_template('index.html')

# Route to handle PDF upload and translation
@app.route('/translate_pdf', methods=['POST'])
def translate_pdf():
    if 'file' not in request.files:
        return jsonify({"error": "No file provided"}), 400
```

Gambar 20 Kutipan kode konfigurasi Flask

Flask digunakan untuk menginisiasi aplikasi web melalui Flask(__name__). *Endpoint* utama didefinisikan dengan @app.route('/'), yang berfungsi untuk menampilkan halaman HTML (index.html). Untuk menangani unggahan *file* PDF dan menerjemahkannya, digunakan @app.route('/translate_pdf', methods=['POST']). Selain itu, request.files['file'] memungkinkan aplikasi mengambil *file* yang diunggah oleh pengguna untuk diproses lebih lanjut.

B. Pymupdf

Pustaka pymupdf digunakan untuk membaca dan mengekstrak teks dari *file* PDF yang diunggah. Penggunaan pymupdf ini untuk mempersingkat waktu pengembangan karena proyek yang mirip dengan kebutuhan penulis suah tersedia. Sehingga tidak diperlukan pengembangan dari awal lagi untuk penyesuaian *use case* yang dibutuhkan. Berikut kutipan kode penggunaan pymupdf pada sistem penerjemah dokumen otomatis.

```
import pymupdf  # Importing pymupdf directly

# Open the PDF with pymupdf
try:
    doc = pymupdf.open(input_path)
    textflags = pymupdf.TEXT_DEHYPHENATE

    # Iterate over all pages for translation
    for page in doc:
        blocks = page.get_text("dict", flags=textflags)["blocks"]

        for block in blocks:
            for line in block["lines"]:
                for span in line["spans"]:
                    bbox = span["bbox"]
```

Gambar 21 Kutipan kode konfigurasi pymupdf

C. Deep-translator

Pustaka ini mendukung berbagai layanan penerjemahan, seperti Google Translate, Microsoft Translator, dan DeepL Dengan menggunakan Deep-Translator, pengguna dapat menerjemahkan teks secara otomatis dalam berbagai bahasa tanpa batasan jumlah terjemahan, sehingga memungkinkan pemrosesan teks yang lebih efisien dan akurat [13]. Dalam sistem penerjemah dokumen berbasis OCR, pustaka ini digunakan bersama pymupdf untuk menerjemahkan teks yang diekstrak dari PDF antara Bahasa Indonesia dan Inggris sesuai kebutuhan pengguna. Berikut adalah kutipan kode penggunaan deep-translator dalam sistem penerjemah dokumen berbasis NLP.

```
from deep_translator import GoogleTranslator
import os

# Inisialisasi penerjemah dari Bahasa Indonesia ke Bahasa Inggris
to_english = GoogleTranslator(source="id", target="en")
```

Gambar 22 Kutipan kode konfigurasi deep-translator

### 3.2.2. Menerjemahkan Dokumen

Setelah konfigurasi pustaka ditentukan, penerapan tiap-tiap pustaka kemudian dilakukan untuk menerapkan proses otomasi penerjemahan pada sistem. Penerjemahan sistem dilakukan menggunakan pustaka deep-translator dari Bahasa Indonesia menjadi Bahasa Inggris. Penerjemahan dilakukan secara blok-blok teks hasil pemrosesan pymupdf. Hasil terjemahan kemudian dilakukan penyusunan kembali menjadi format PDF semula. Berikut merupakan kutipan kode proses penerjemahan dokumen.

```
import pymupdf  # Importing pymupdf directly
from deep_translator import GoogleTranslator

# Define translator and white color
WHITE = pymupdf.pdfcolor["white"]
to_english = GoogleTranslator(source="id", target="en")

# Open the PDF with pymupdf
doc = pymupdf.open("input_document.pdf")
textflags = pymupdf.TEXT_DEHYPHENATE

# Create Optional Content layer named "Indonesian" and activate it
ocg_xref = doc.add_ocg("Indonesian", on=True)

# Subset fonts and save the output file
doc.subset_fonts()
doc.ez_save("translated_document.pdf")
doc.close()
```

Gambar 23 Kutipan kode penerjemahan dokumen

### 3.2.3. Pengiriman Dokumen Hasil Terjemah

Setelah penerjemahan selesai, dilakukan konfigurasi *endpoint* untuk mengintegrasikan sistem penerjemah dokumen berbasis NLP dengan aplikasi arsip digital. Proses ini membungkus seluruh alur penerjemahan dalam Flask, memungkinkan pemrosesan unggahan dokumen melalui *endpoint* POST. Berikut adalah kutipan kode konfigurasi *endpoint* menggunakan Flask.

```python
from flask import Flask, request, send_file, jsonify
import os

# Initialize Flask app
app = Flask(__name__)

# Route to handle PDF upload and translation
@app.route('/translate_pdf', methods=['POST'])
def translate_pdf():
    if 'file' not in request.files:
        return jsonify({"error": "No file provided"}), 400

    file = request.files['file']

    # Check if the file is a PDF
    if not file.filename.endswith('.pdf'):
        return jsonify({"error": "File is not a PDF"}), 400

    # Save the uploaded PDF temporarily
    input_path = "input_document.pdf"
    output_path = "translated_document.pdf"
    file.save(input_path)

    try:
        # Call the function that processes the PDF and translates the text
        process_pdf(input_path, output_path)
    except Exception as e:
        return jsonify({"error": f"Failed to process the PDF: {str(e)}"}), 500

    # Return the translated PDF to the client
    return send_file(output_path, as_attachment=True)

# Run the Flask app
if __name__ == '__main__':
    app.run(debug=True)
```

Gambar 24 Kutipan kode pengiriman hasil terjemah

## 3.3 Evaluasi Sistem OCR

### 3.3.1. Evaluasi Akurasi Ekstraksi

Pengujian dilakukan menggunakan *input* berupa dokumen asli yang diberikan oleh PT NRC. Namun, karena keterbatasan jumlah dokumen yang diberikan akses oleh PT NRC untuk digunakan dalam proses pengembangan, pengujian performa sistem secara keseluruhan tidak dapat dilakukan dengan baik. Sebagai alternatif, pengujian dilakukan dengan tiga skenario untuk tiap-tiap jenis dokumen yang digunakan, yaitu dokumen berbentuk format *native* PDF, dokumen hasil *scan* menggunakan printer, dan dokumen hasil *scan* menggunakan *scanner handphone*.

Seluruh pengujian menggunakan ketiga metode berupa *Levenshtein similarity, word accuracy,* dan *cosine similarity* dilakukan secara otomatis menggunakan dukungan pustaka scikit-learn yang tersedia dalam bahasa pemrograman python.

A. Levenshtein Similarity

Levenshtein Similarity adalah metrik berbasis Levenshtein Distance, yang digunakan untuk mengukur tingkat kesamaan antara dua string dengan menghitung jumlah minimum operasi edit (penyisipan, penghapusan, atau substitusi karakter) yang diperlukan untuk mengubah satu *string* menjadi *string* lainnya. Dalam konteks evaluasi akurasi hasil *Optical Character Recognition* (OCR), *Levenshtein Similarity* digunakan untuk membandingkan teks yang diekstraksi oleh sistem OCR dengan teks referensi yang benar [14].

$$Lv\ Similarity = \left(\frac{Lv\ Distance\ (A,B)}{1-\max(|A|,|B|)}\right) \times 100\% \qquad (1)$$

Dengan A dan B adalah teks ekstraksi dan referensi. Berikut adalah hasil pengujian yang dilakukan.

Tabel 1 Evaluasi akurasi ekstraksi OCR dengan *Levenshtein Similarity*

| Jenis Dokumen | Akurasi | | |
|---|---|---|---|
| | ***Scan Printer*** | ***Scan* HP** | **Native PDF** |
| CV | 98,16% | 97,99% | 99,61% |
| Tenaga Ahli | 83,76% | 67,60% | 87,41% |
| Surat Masuk | 91,01% | 88,20% | - |
| Surat Keluar | 95,73% | 93,30% | - |
| Kontrak | 99,11% | 87,24% | - |
| Legalitas | 87,92% | 71,42% | 90,99% |
| Keuangan | 94,32% | 75,11% | 77,00% |
| Pengurus | 93,40% | - | - |
| Pemegang Saham | 97,68% | - | - |
| Akurasi rata-rata | 93,45% | 82,98% | 88,75% |

B. *Word Accuracy*

*Word Accuracy* (WA) adalah metrik yang digunakan untuk mengevaluasi seberapa akurat sistem *Optical Character Recognition* (OCR) dalam mengenali kata-kata dalam suatu teks dibandingkan dengan teks referensi [15].

$$Word\ Accuracy = \left(\frac{N-S-D-I}{N}\right) \times 100\% \qquad (2)$$

Dengan N adalah jumlah kata salah dalam teks, S kata salah berupa substitusi, D penghapusan, dan I tambahan. Berikut adalah hasil pengujian yang telah dilakukan

Tabel 2 Evaluasi akurasi ekstraksi OCR dengan *Word Accuracy*

| Jenis Dokumen | Akurasi | | |
|---|---|---|---|
| | ***Scan Printer*** | ***Scan* HP** | ***Native* PDF** |
| CV | 98,16% | 97,99% | 99,61% |
| Tenaga Ahli | 83,76% | 67,60% | 87,41% |
| Surat Masuk | 91,01% | 88,20% | - |
| Surat Keluar | 95,73% | 93,30% | - |
| Kontrak | 99,11% | 87,24% | - |
| Legalitas | 87,92% | 71,42% | 90,99% |
| Keuangan | 94,32% | 75,11% | 77,00% |
| Pengurus | 93,40% | - | - |
| Pemegang Saham | 97,68% | - | - |
| Akurasi rata-rata | 93,45% | 82,98% | 88,75% |

C. *Cosine Similarity*

*Cosine similarity* adalah metrik yang digunakan untuk mengukur kesamaan antara dua vektor dalam ruang multidimensi berdasarkan sudut kosinus di antara keduanya. Pendekatan ini bekerja dengan merepresentasikan teks sebagai vektor berdasarkan frekuensi kemunculan kata atau karakter, kemudian menghitung sudut kosinus antara kedua vektor tersebut. Semakin kecil sudutnya, semakin mirip kedua teks tersebut, dengan nilai berkisar antara 0 hingga 1 [15]. Dalam konteks evaluasi OCR, metode ini digunakan untuk mengukur seberapa mirip teks hasil ekstraksi OCR dengan teks referensi.

$$cos(\theta) = \frac{A \cdot B}{\|A\| \times \|B\|} \quad (3)$$

Dengan A dan B adalah vektor teks hasil ekstraksi dan teks referensi. Berikut hasil pengujian yang telah dilakukan.

Tabel 3 Evaluasi akurasi ekstraksi OCR dengan *Cosine Similarity*

| Jenis Dokumen | Akurasi | | |
|---|---|---|---|
| | ***Scan Printer*** | ***Scan* HP** | ***Native* PDF** |
| CV | 98,16% | 97,99% | 99,61% |
| Tenaga Ahli | 83,76% | 67,60% | 87,41% |
| Surat Masuk | 91,01% | 88,20% | - |
| Surat Keluar | 95,73% | 93,30% | - |
| Kontrak | 99,11% | 87,24% | - |
| Legalitas | 87,92% | 71,42% | 90,99% |
| Keuangan | 94,32% | 75,11% | 77,00% |
| Pengurus | 93,40% | - | - |
| Pemegang Saham | 97,68% | - | - |
| Akurasi rata-rata | 93,45% | 82,98% | 88,75% |

## 3.3.2. Evaluasi Hasil Ekstraksi Regex

Mekanisme pengujian dilakukan dengan mengirimkan dokumen melalui HTTP POST dengan body yang berisi parameter jenis dokumen dan *file* PDF yang sesuai dengan jenis dokumen tersebut. Pengujian dilakukan menggunakan bantuan aplikasi Postman untuk mengirimkan permintaan ke *endpoint* sub sistem OCR

Tabel 4 Evaluasi integrasi Sistem OCR

| Jenis Dokumen | Ekstraksi Benar | Ekstraksi Salah | Total | Akurasi |
|---|---|---|---|---|
| CV | 33 | 0 | 33 | 100,00% |
| Tenaga Ahli | 15 | 0 | 15 | 100,00% |
| Surat Masuk | 10 | 2 | 12 | 83,33% |
| Surat Keluar | 7 | 2 | 9 | 77,78% |
| Kontrak | 7 | 1 | 8 | 87,50% |
| Legalitas | 9 | 3 | 12 | 75,00% |
| Keuangan | 10 | 2 | 12 | 83,33% |
| Pengurus | 16 | 0 | 16 | 100,00% |
| Pemegang Saham | 12 | 0 | 12 | 100,00% |
| Akurasi Rata-Rata | | | | 89,65% |

## 3.3.3. Evaluasi Integrasi Sistem OCR

Pengujian ini dilakukan untuk menilai keandalan sistem ekstraksi dokumen otomatis berbasis OCR dalam beroperasi. Pengujian dilakukan dengan menghitung lama waktu pemrosesan dokumen oleh sistem dan tingkat kegagalan sistem dalam menjalankan proses ekstraksi dokumen. Mekanisme pengujian dilakukan dengan mengirimkan dokumen melalui HTTP POST dengan aplikasi Postman ke *endpoint* sub sistem OCR.

Tabel 5 Evaluasi integrasi Sistem OCR

| Jenis Dokumen | Ekstraksi Berhasil | Ekstraksi Gagal | Waktu Pemrosesan Rata-rata (detik) |
|---|---|---|---|
| CV | 10 | 0 | 4,06 |
| Tenaga Ahli | 10 | 0 | 8,38 |
| Surat Masuk | 10 | 0 | 4,36 |
| Surat Keluar | 10 | 0 | 3,98 |
| Kontrak | 10 | 0 | 12,25 |
| Legalitas | 10 | 0 | 16,69 |
| Keuangan | 10 | 0 | 17,47 |
| Pengurus | 10 | 0 | 4,33 |
| Pemegang Saham | 10 | 0 | 4,18 |

## 3.4 Evaluasi Sistem NLP

Pengujian yang dilakukan bertujuan untuk menilai performa sistem penerjemah otomatis berbasis NLP yang dikembangkan dilakukan dengan menilai hasil akurasi terjemahan mesin yang digunakan pada sistem, yaitu deep-translator menggunakan beberapa *dataset* yang tersedia dan dapat menunjukkan performa penerjemahan dari Bahasa Indonesia ke Bahasa Inggris. Pada pengujian ini, *dataset* yang digunakan adalah *dataset* NusaX-MT dengan metrik pengukuran menggunakan BLEU, METEOR, TER, dan chrF.

Tabel 6 Evaluasi akurasi terjemahan sistem

| Metrik | Nilai |
|---|---|
| BLEU | 0,128 |
| METEOR | 0,420 |
| Translation Edit Rate (TER) | 65,435 |
| chrF | 48,411 |

# 4. Kesimpulan

Sistem ekstraksi dokumen otomatis berbasis OCR dirancang menggunakan Flask, pdf2image, dan OpenCV, dengan Tesseract OCR untuk ekstraksi teks secara otomatis. Berbagai teknik pra-pemrosesan seperti

*deskewing*, *adaptive thresholding*, dan *noise reduction* telah dioptimalkan untuk meningkatkan akurasi hasil ekstraksi. Ekstraksi informasi penting menggunakan regex telah dikembangkan untuk berbagai jenis dokumen, sementara sistem penerjemahan otomatis berbasis NLP menggunakan pymupdf dan deep-translator mampu menerjemahkan dokumen dengan tetap mempertahankan struktur teks. API berbasis Flask telah dibangun untuk mendukung integrasi dengan aplikasi arsip berbasis web. Dengan sistem ini, proses ekstraksi dan penerjemahan dokumen menjadi lebih efisien serta dapat dikembangkan lebih lanjut dengan alternatif *engine* OCR, fine-tuning parameter, dan integrasi LLM untuk pemahaman dokumen yang lebih baik.

## Referensi


[1] M. I. Anshari and R. Manjaleni, "Pengaruh Digitalisasi Terhadap Efisiensi dan Efektivitas Proses Akuntansi Pada Koperasi Pesantren," *ARBITRASE: Journal of Economics and Accounting*, vol. 5, no. 1, pp. 51–58, Jul. 2024, doi: 10.47065/arbitrase.v5i1.2036.

[2] P. Goswami, N. Faujdar, S. Debnath, A. K. Khan, and G. Singh, "Investigation on storage level data integrity strategies in cloud computing: classification, security obstructions, challenges and vulnerability," Dec. 01, 2024, *Springer Science and Business Media Deutschland GmbH*. doi: 10.1186/s13677-024-00605-z.

[3] A. Anisah, D. Wahyuningsih, E. Helmud, T. Suwanda, P. Romadiana, and D. Irawan, "Rancang Bangun Sistem Informasi Manajemen Arsip Digital," *Jurnal Sisfokom (Sistem Informasi dan Komputer)*, vol. 10, no. 3, pp. 419–425, Dec. 2021, doi: 10.32736/sisfokom.v10i3.1300.

[4] M. Nafsin, A. Qashlim, and U. Khairat, "SISTEM INFORMASI DATA SISWA BERBASIS OCR (OPTICAL CHARACTER RECOGNITION) PADA SMK BINA HARAPAN," *Journal Peqguruang: Conference Series*, vol. 4, no. 1, p. 412, May 2022, doi: 10.35329/jp.v4i1.2201.

[5] R. D. S, S. S. Nagar, R. S. Upendra, and R. Karthik, "Design and Development of User-friendly Bi-lingual Translation System Employing Machine Translation 5 Base Deep Learning Neural Network Framework Based NLP," in *2024 2nd International Conference on Artificial Intelligence and Machine Learning Applications Theme: Healthcare and Internet of Things (AIMLA)*, IEEE, Mar. 2024, pp. 1–6. doi: 10.1109/AIMLA59606.2024.10531504.

[6] K. Apriyanti and T. Wahyu Widodo, "Implementasi Optical Character Recognition Berbasis Backpropagation untuk Text to Speech Perangkat Android," *IJEIS (Indonesian Journal of Electronics and Instrumentation Systems)*, vol. 6, no. 1, p. 13, Apr. 2016, doi: 10.22146/ijeis.10767.

[7] S. Al Sowaidi, O. Isa, and O. P. Alam, "Structured Document Digitalization System," *International Journal of Computing and Digital Systems*, vol. 10, pp. 1–6, 2020.

[8] G. Waja, J. Shah, and P. Nanavati, "AGILE SOFTWARE DEVELOPMENT," *International Journal of Engineering Applied Sciences and Technology*, vol. 5, no. 12, Apr. 2021, doi: 10.33564/IJEAST.2021.v05i12.011.

[9] Nur Aeni Hidayah and N. Rofiqoh, "EVALUASI SOFTWARE VISUAL STUDIO CODE MENGGUNAKAN METODE QUETIONNAIRES NELSEN'S ATTRIBUTES OF USABILITY (NAU)," *JURNAL PERANGKAT LUNAK*, vol. 6, no. 3, pp. 382–391, Oct. 2024, doi: 10.32520/jupel.v6i3.3383.

[10] E. Ferdiana Sari, "PENERAPAN GITHUB SEBAGAI MEDIA E-LEARNING UNTUK MENGETAHUI KEEFEKTIFAN KOLABORASI PROJECT PADA MATA PELAJARAN PEMROGRAMAN WEB DAN PERANGKAT BERGERAK DI SMK NEGERI 2 SURABAYA," *Jurnal IT-EDU*, vol. 06, pp. 14–22, 2021.

[11] P. P. Kore, M. J. Lohar, M. T. Surve, and S. Jadhav, "API Testing Using Postman Tool," *Int J Res Appl Sci Eng Technol*, vol. 10, no. 12, pp. 841–843, Dec. 2022, doi: 10.22214/ijraset.2022.48030.

[12] L. Albesher and R. Alfayez, "An Observational Study on Flask Web Framework Questions on Stack Overflow (SO)," *IET Software*, vol. 2024, no. 1, Jan. 2024, doi: 10.1049/sfw2/1905538.

[13] N. K. R. Sari, I. M. A. D. Suarjaya, and P. W. Buana, "Perbandingan Translation Library Pada Python (Studi Kasus: Analisis Sentimen Penyakit Menular Di Indonesia)," *2021 JITTER-Jurnal Ilmiah Teknologi dan Komputer*, vol. 2, no. 3, pp. 1–7, 2021.

[14] P. Janardhana Rao, K. Nageswara Rao, S. Gokuruboyina, and K. N. Neeraja, "An Efficient Methodology for Identifying the Similarity Between Languages with Levenshtein Distance," 2024, pp. 161–174. doi: 10.1007/978-981-99-7137-4_15.

[15] T. W. Ramdhani, I. Budi, and B. Purwandari, "Optical Character Recognition Engines Performance Comparison in Information Extraction," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 8, 2021, doi: 10.14569/IJACSA.2021.0120814.

## Biodata Penulis

Fajrul Kamal (21060120130082). Lahir di Kabupaten Pekalongan pada tanggal 15 September 2002. Telah menempuh pendidikan mulai dari MIS Pakumbulan selama 6 tahun, MTsS Simbangkulon 1 selama 3 tahun, serta MA Salafiyah Simbangkulon selama 3 tahun. Saat ini penulis sedang menempuh pendidikan sarjana S-1 Teknik Elektro di Fakultas Teknik, Universitas Diponegoro, Semarang dengan konsentrasi Teknologi Informasi angkatan 2020.

Saya menyatakan bahwa segala informasi yang tersedia di makalah ini adalah benar, merupakan hasil karya sendiri, dan semua karya orang lain telah dikutip dengan benar.

**Fajrul Kamal**
21060120130082

**Pengesahan**
Telah disetujui untuk diajukan pada Seminar Tugas Akhir.

Pembimbing I

**Ir. Aghus Sofwan, S.T., M.T., Ph.D., IPU.**
NIP. 197302041997021001

Pembimbing II

**Yuli Christyono, S.T., M.T.**
NIP. 196807111997021001