**Zusammenfassung: Automated Document Management Workflow**

**Projekt/Problemstellung**

Entwicklung eines verbesserten Python-Skripts zur automatischen Verarbeitung von PDF-Dokumenten. Das Programm analysiert PDFs mit Hilfe von OpenAI's KI, extrahiert relevante Informationen und benennt die Dateien nach einem einheitlichen Schema um. Es erkennt Duplikate, behandelt Fehler und sortiert verarbeitete Dokumente in entsprechende Ordner.

**Projektaufbau und Implementierung**

**Dateien im Projekt**

1. autodocs\_v2.py - Hauptskript mit der gesamten Programmlogik
2. duplicate\_detector.py - Hilfsklasse zur Erkennung von Dokumentduplikaten
3. .env - Enthält den OpenAI API-Schlüssel (OPENAI\_API\_KEY=dein\_api\_key\_hier)
4. autodocs\_config.yaml - Konfigurationsdatei für alle Programmeinstellungen

**Ordnerstruktur**

* Basisordner: C:\Users\renem\OneDrive\09\_AutoDocs\
  + 01\_InboxDocs\ - Für eingehende PDF-Dokumente
  + 02\_FinalDocs\ - Für verarbeitete, umbenannte Dokumente
  + 03\_TrashDocs\ - Für fehlerhafte oder problematische Dokumente

**Installierte Pakete**

Alle erforderlichen Python-Pakete wurden mit folgenden Befehlen installiert:

pip install pymupdf python-dotenv openai pyyaml

**Wichtige Verbesserungen im Code**

1. **Robustere PDF-Textextraktion** mit spezifischer Fehlerbehandlung für verschiedene PDF-Probleme
2. **Verbesserte OpenAI-API-Anbindung** mit Wiederholungsversuchen bei Fehlern
3. **Erweiterte Dateinamenvalidierung** mit Plausibilitätsprüfungen für Daten
4. **Zweistufige Duplikaterkennung** mittels Hash- und Inhaltsvergleich
5. **Detaillierte Prozessberichterstattung** mit JSON-Verarbeitungsbericht
6. **YAML-Konfigurationssystem** für einfache Anpassungen ohne Codeänderungen
7. **Kommandozeilenoptionen** für flexible Nutzung

**Wichtige Befehle und Eingaben**

**Ausführen des Skripts**

* Standard-Ausführung: python autodocs\_v2.py
* Simulationsmodus (keine Änderungen): python autodocs\_v2.py --dry-run
* Mit erhöhtem Detaillierungsgrad: python autodocs\_v2.py -v oder python autodocs\_v2.py -vv
* Einzelne Datei verarbeiten: python autodocs\_v2.py --single-file "C:\Pfad\zu\deiner\datei.pdf"
* Konfiguration zurücksetzen: python autodocs\_v2.py --rebuild-config
* Vorhandene Dateien überschreiben: python autodocs\_v2.py --force

**Konfigurationsmöglichkeiten in YAML**

Die wichtigsten Konfigurationsoptionen in autodocs\_config.yaml:

yaml

paths:

input\_dir: C:\Users\renem\OneDrive\09\_AutoDocs\01\_InboxDocs

output\_dir: C:\Users\renem\OneDrive\09\_AutoDocs\02\_FinalDocs

trash\_dir: C:\Users\renem\OneDrive\09\_AutoDocs\03\_TrashDocs

openai:

model: gpt-3.5-turbo *# Alternativ: gpt-4o für höhere Präzision*

temperature: 0.3

max\_retries: 3

document\_processing:

max\_file\_size\_mb: 20

similarity\_threshold: 0.85

valid\_doc\_types:

- rechnung

- vertrag

- brief

- meldung

- bescheid

- dokument

- antrag

**Erkenntnisse / Entscheidungen**

* **KI-Modell**: Beginn mit GPT-3.5-Turbo für Kosteneffizienz, bei Bedarf Upgrade auf GPT-4o für höhere Präzision
* **Nutzungsmethoden**:
  1. Manuell über Windows Explorer
  2. Desktop-Verknüpfung
  3. Automatisierung über Windows-Aufgabenplanung
  4. Kommandozeile für spezifische Funktionen
* **Duplikaterkennung**: Kombination aus Hash-basierter und inhaltsbasierter Ähnlichkeitsanalyse (Schwellenwert: 85%)
* **Konfigurierbarkeit**: Alle wichtigen Parameter wurden in die YAML-Datei ausgelagert, um Anpassungen ohne Codeänderungen zu ermöglichen

**Offene Punkte / Nächster Schritt**

1. **Testen mit realen Dokumenten**: Zunächst im Simulationsmodus (--dry-run) prüfen
2. **Modellentscheidung**: Bei Bedarf von GPT-3.5-Turbo auf GPT-4o umsteigen
3. **Automatisierung**: Nach erfolgreicher Testphase Einrichtung der automatischen Ausführung über die Windows-Aufgabenplanung
4. **Konfiguration anpassen**: Bei Bedarf Anpassung der Dokumenttypen und anderer Parameter in der YAML-Datei

**Stand**

Das Programm ist vollständig implementiert und einsatzbereit. Alle notwendigen Dateien sind erstellt und die erforderlichen Bibliotheken installiert. Der nächste Schritt ist das Testen mit realen Dokumenten.