**Zusammenfassung: Automated Document Management Workflow**

**Projekt/Problemstellung**

Wir haben ein Python-Programm zur automatischen Verarbeitung von PDF-Dokumenten verbessert. Das Skript analysiert PDFs mittels OpenAI's KI, extrahiert Informationen, benennt Dateien nach einem einheitlichen Schema um und sortiert sie in entsprechende Ordner. Hauptziel des aktuellen Chats war die Erstellung einer benutzerfreundlichen GUI (graphische Benutzeroberfläche) für das Programm sowie die Integration zusätzlicher Funktionen zur Duplikaterkennung.

**Projektaufbau**

**Dateien im Projekt**

* autodocs\_v2.py - Hauptskript mit der Programmlogik
* duplicate\_detector.py - Hilfsklasse zur Erkennung von Dokumentduplikaten
* .env - Enthält den OpenAI API-Schlüssel
* autodocs\_config.yaml - Konfigurationsdatei
* autodocs\_gui.py - Neu erstellte GUI-Datei
* maehrdocs.bat - Batchdatei zum einfachen Starten der GUI

**Ordnerstruktur**

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

**Installierte Pakete**

pip install pymupdf python-dotenv openai pyyaml

**Wichtige Befehle und Eingaben**

**Code-Erweiterungen**

Für die Duplikaterkennung wurde der DuplicateDetector erweitert:

python

class DuplicateDetector:

def \_\_init\_\_(self):

*# Deutsche Stopwörter definieren*

self.STOPWORDS = {'der', 'die', 'das', 'und', 'mit', 'von', 'für', 'ist', 'ein', 'eine'}

def calculate\_similarity(self, text1, text2):

"""Berechnet die Textähnlichkeit zwischen zwei Dokumenten"""

*# Texte in Wörter aufteilen*

words1 = set(self.\_tokenize(text1))

words2 = set(self.\_tokenize(text2))

*# Leere Texte vermeiden*

if not words1 or not words2:

return 0.0

*# Gemeinsame Wörter zählen*

common\_words = words1.intersection(words2)

*# Jaccard-Ähnlichkeit berechnen*

similarity = len(common\_words) / (len(words1) + len(words2) - len(common\_words))

return similarity

def \_tokenize(self, text):

"""Text in Wörter aufteilen und bereinigen"""

if not text:

return []

*# Alles klein schreiben und Sonderzeichen entfernen*

text = text.lower()

import re

*# Nur Wörter mit mind. 3 Buchstaben behalten und Stopwörter entfernen*

words = [w for w in re.findall(r'\b[a-z]{3,}\b', text) if w not in self.STOPWORDS]

return words

**GUI-Entwicklung**

Wir haben eine dunkle, moderne GUI mit tkinter erstellt. Der vollständige Code wurde in eine Datei namens autodocs\_gui.py gespeichert mit einem angepassten Farbschema:

python

self.colors = {

"background\_dark": "#0D1117", *# Tiefdunkles Blau-Schwarz – idealer Dark-Mode-Hintergrund*

"background\_medium": "#161B22", *# Etwas hellerer Hintergrund für Sektionen und Abgrenzungen*

"card\_background": "#1F2937", *# Dunkles Grau-Blau für Panels/Karten – klar vom Hintergrund abgesetzt*

"primary": "#3B82F6", *# Auffälliges, modernes Blau für Buttons und Hauptaktionen*

"accent": "#60A5FA", *# Helleres Blau für Hover-Effekte, Hervorhebungen oder sekundäre Aktionen*

"text\_primary": "#F9FAFB", *# Fast weiß – maximaler Kontrast auf dunklem Hintergrund*

"text\_secondary": "#9CA3AF", *# Mittelhelles Grau-Blau für unaufdringlichen, gut lesbaren Sekundärtext*

"success": "#10B981", *# Frisches Grün für Erfolgsmeldungen oder Statusanzeigen*

"warning": "#FBBF24", *# Sattes Gelb-Orange für Warnhinweise oder Hinweise mit hoher Priorität*

"error": "#EF4444" *# Kräftiges Rot für Fehlermeldungen oder kritische Hinweise*

}

**Startmethode mit Batch-Datei**

Um die GUI einfach zu starten, wurde eine Batchdatei erstellt:

@echo off

start pythonw autodocs\_gui.py

Diese Datei wurde als maehrdocs.bat im Projektordner gespeichert.

**Erkenntnisse / Entscheidungen**

1. **GUI-Design**: Wir haben uns für ein dunkles, modernes Design mit dem Namen "maehrdocs" entschieden.
2. **Fenstergröße**: Die GUI wurde auf eine Größe von 1700x1300 Pixeln angepasst, um alle Elemente übersichtlich darzustellen.
3. **Startmethode**: Anstatt eine komplexe .exe-Datei mit PyInstaller zu erstellen, wurde die einfachere Lösung einer Batch-Datei gewählt.
4. **Duplikaterkennung**: Die Duplikaterkennung wurde verbessert, indem Stopwörter aus der Ähnlichkeitsberechnung ausgeschlossen wurden.
5. **Benutzerfreundlichkeit**: Die GUI bietet ein übersichtliches Dashboard mit Status der Verzeichnisse, Protkollausgabe und verschiedenen Funktionen.

**Offene Punkte / Nächster Schritt**

1. **Testen der GUI** mit realen Dokumenten.
2. **Anpassung der YAML-Konfiguration** bei Bedarf.
3. **Optimierung des Workflows** durch Benutzerrückmeldungen.
4. **Überprüfung der Duplikaterkennung** in der Praxis, insbesondere die Wirksamkeit der Stopwörter.
5. **Verknüpfung der GUI** für einfachen Zugriff (z.B. auf dem Desktop).

**Stand: Mai 2025**

Das Programm ist vollständig implementiert und einsatzbereit. Die neu erstellte GUI bietet eine benutzerfreundliche Oberfläche für das automatische Dokumentenmanagement. Die GUI kann über die Batch-Datei maehrdocs.bat gestartet werden und zeigt den aktuellen Status der verschiedenen Dokumentenordner an.