Automatisierte Textanalyse und Dokumentenverarbeitung

Effiziente Lösungen mit analyse cmd.py und aictrl2.py

Herbert Dirnberger

26. August 2024

Executive Summary Diese Dokumentation präsentiert die Entwicklung und Anwendung der Tools analyse cmd.py und aictrl2.py, die entwickelt wurden, um Textanalyse- und Dokumentenverarbeitungsprozesse zu automatisieren. Das Hauptziel des Projekts ist es, manuelle Prozesse effizienter zu gestalten, indem große Mengen an Texten aus verschiedenen Dokumentenformaten extrahiert, analysiert und in strukturierte Berichte oder neue Inhalte umgewandelt werden. Die Dokumentation liefert praxisnahe Beispiele, technische Einblicke und zeigt die Anpassungsfähigkeit der Tools in unterschiedlichen Branchen auf.

Die Zielgruppe für diese Arbeit umfasst IT-Entwickler, Rechtsanwälte, Compliance-Experten, Projektmanager, Unternehmensberater und Entscheidungsträger, die auf der Suche nach effizienten Automatisierungslösungen sind. Diese Tools sind sowohl für technische als auch nicht-technische Fachkräfte relevant, die ihre Arbeitsprozesse durch den Einsatz von Python und KI-Technologien verbessern möchten.

Der Einsatz von analyse cmd.py und aictrl2.py bietet signifikante Vorteile: Es werden Zeit- und Kostenersparnisse durch die Automatisierung erzielt, die Fehlerquote wird reduziert und die Qualität der Ergebnisse steigt. Zudem bieten die Tools konsistente und flexible Workflows, die sich an unterschiedliche Anforderungen anpassen lassen. Diese Effizienz macht sie besonders wertvoll in Projekten mit hohen Datenvolumen und komplexen Anforderungen. Insgesamt tragen diese Tools entscheidend zur Optimierung von Dokumentenprozessen bei.

Inhaltsverzeichnis

1 Einführung

Diese Dokumentation beschreibt die Entwicklung und den Einsatz der Tools `analyse cmd.py` und `aictrl2.py` zur automatisierten Textanalyse und Dokumentenverarbeitung. Diese beiden Tools wurden entwickelt, um große Mengen an Dokumenten effizient zu verarbeiten, zu analysieren und daraus strukturierte Berichte oder neue Inhalte zu generieren. Ziel ist es, manuelle Prozesse zu automatisieren, Zeit zu sparen und die Qualität der Dokumente zu verbessern.

1.1 Projektziele

Das Hauptziel des Projekts (Proof of Concepts) ist die Entwicklung einer umfassenden Lösung für die automatisierte Verarbeitung und Analyse von Textdokumenten. Durch die Kombination der beiden Tools sollen folgende Ziele erreicht werden:

- **Effizienzsteigerung:** Automatisierung der Dokumentenverarbeitung, um manuelle Arbeit zu minimieren und Fehler zu reduzieren.
- **Flexibilität:** Anpassungsfähige Tools, die in verschiedenen Branchen und für unterschiedliche Anwendungsfälle eingesetzt werden können.
- **Qualitätsverbesserung:** Sicherstellen, dass die verarbeiteten Dokumente konsistent, vollständig und frei von Fehlern sind.
- **Skalierbarkeit:** Möglichkeit, große Mengen an Dokumenten gleichzeitig zu verarbeiten.

1.2 Entscheidung für Python statt einer Low-Code-Plattform

Die Entscheidung Python als Programmiersprache für die Entwicklung der Tools `analyse cmd.py` und `aictrl2.py` zu verwenden, anstatt eine Low-Code-Plattform wie make.com zu nutzen. Diese Wahl war strategisch und auf die spezifischen Anforderungen des Projekts abgestimmt.

- **Flexibilität und Kontrolle**: Python bietet eine außergewöhnliche Flexibilität und Kontrolle über den gesamten Entwicklungsprozess. Während Low-Code-Plattformen durch vorgegebene Funktionen und Module eingeschränkt sein können, erlaubt Python eine maßgeschneiderte Entwicklung, die speziell auf die komplexen Anforderungen dieses Projekts zugeschnitten ist. Die Möglichkeit, jeden Aspekt des Codes zu gestalten und anzupassen, war entscheidend, um die spezifischen Anforderungen wie die Integration der OpenAI-APIs und die fortgeschrittene Verarbeitung von Texten und Platzhaltern vollständig umzusetzen.
- **Integration komplexer Bibliotheken**: Ein zentraler Aspekt dieses Projekts war die nahtlose Integration von OpenAI-APIs, die erweiterte Funktionalitäten wie die Verarbeitung und Analyse von Texten ermöglichen. Solche spezifischen Anforderungen wären in einer Low-Code-Umgebung schwer umzusetzen oder könnten

überhaupt nicht realisiert werden. Python hingegen verfügt über eine umfangreiche Bibliothekslandschaft, die die Entwicklung spezialisierter Funktionen erheblich erleichtert.

- **Wiederverwendbarkeit und Anpassungsfähigkeit**: Die Möglichkeit, Code einfach zu erweitern, anzupassen und wiederzuverwenden, war ein weiterer Grund für die Wahl von Python. Diese Wiederverwendbarkeit ist besonders wichtig, da das Tool auch für zukünftige Projekte genutzt werden soll, wie beispielsweise die Ausführung von Large Language Models (LLMs) wie LLaMA3 auf eigener Hardware. Python ermöglicht es, den entwickelten Code flexibel an verschiedene Hardwareanforderungen und Anwendungsfälle anzupassen.
- **Leistung und Effizienz**: Python bietet die notwendigen Werkzeuge, um leistungsstarke Algorithmen und Prozesse effizient zu implementieren. Bei der Verarbeitung großer Textmengen oder komplexer Dokumentenstrukturen sind Leistung und Effizienz von entscheidender Bedeutung. Python erlaubt es, Prozesse zu optimieren, die in einer Low-Code-Umgebung möglicherweise langsamer oder weniger effizient wären.
- **Skalierbarkeit und Erweiterbarkeit**: Ein weiterer Vorteil von Python ist die einfache Skalierbarkeit des Projekts auf größere Datenmengen und komplexere Anwendungsfälle. Während Low-Code-Plattformen in bestimmten Szenarien hilfreich sein können, stoßen sie bei der Verarbeitung großer Datenmengen oder der Integration komplexer Algorithmen häufig an ihre Grenzen. Python bietet die notwendige Skalierbarkeit, um wachsenden Anforderungen gerecht zu werden.
- **Zukunftssicherheit**: Python ist eine der am weitesten verbreiteten Programmiersprachen mit einer großen Entwicklergemeinschaft und umfangreichen Ressourcen. Dies stellt sicher, dass das Projekt auch in Zukunft leicht gepflegt und weiterentwickelt werden kann. Im Vergleich dazu könnten Low-Code-Plattformen in ihrer Entwicklung und ihrem Support eingeschränkt sein, was ihre langfristige Nutzung unsicherer macht.
- **Integration mit bestehenden Systemen**: Die Notwendigkeit, die entwickelten Tools mit bereits bestehenden Systemen und Datenquellen zu integrieren, war ein weiterer Faktor für die Wahl von Python. Die umfangreichen Integrationsmöglichkeiten von Python mit bestehenden Systemen, Datenbanken und APIs machen es zur idealen Wahl, während Low-Code-Plattformen oft nur eingeschränkte Integrationsmöglichkeiten bieten.

Die Entscheidung, Python und nicht eine Low-Code-Plattform zu nutzen, basierte auf der Notwendigkeit, eine flexible, skalierbare und leistungsfähige Lösung zu entwickeln, die sich nahtlos in bestehende Systeme integrieren lässt. Python bietet die erforderliche Freiheit und Kontrolle, um spezialisierte Tools zu entwickeln, die komplexe Anwendungsfälle abdecken und gleichzeitig die Möglichkeit bieten, in neue Anwendungsbereiche zu expandieren. Diese Entscheidung stellt sicher, dass die entwickelten Tools sowohl den aktuellen Anforderungen als auch den zukünftigen Herausforderungen gerecht werden

können.

1.3 Entscheidung für die Modelle GPT-4o und GPT-4o-mini

ie Entscheidung für diese Modelle GPT-4o und GPT-4o-mini basiert auf einer Abwägung zwischen Leistungsfähigkeit, Kosten und spezifischen Anforderungen des Projekts.

- **Leistungsfähigkeit von GPT-4o**: Das Modell GPT-4o wurde gewählt, weil es eine hohe Leistungsfähigkeit und Genauigkeit in der Verarbeitung natürlicher Sprache bietet. Es ist in der Lage, komplexe Textanalysen durchzuführen, präzise Antworten zu generieren und dabei auch kontextuelle Feinheiten zu berücksichtigen. Diese Eigenschaften sind besonders wichtig für Aufgaben wie die Vertragsanalyse, die Erstellung von rechtlichen Gutachten und die Prüfung komplexer Dokumentationen, bei denen eine hohe Genauigkeit erforderlich ist.
- **Ressourcenschonung mit GPT-4o-mini**: Das Modell GPT-4o-mini wurde ergänzend verwendet, um weniger komplexe Aufgaben zu bewältigen, bei denen die volle Leistung von GPT-4o nicht erforderlich ist. GPT-4o-mini bietet eine effizientere Nutzung von Rechenressourcen, wodurch Kosten gesenkt und die Verarbeitungszeit reduziert werden können. Dies ist besonders vorteilhaft für Aufgaben, bei denen eine schnelle Verarbeitung von weniger anspruchsvollen Prompts erforderlich ist, wie zum Beispiel das Erstellen von einfachen Zusammenfassungen oder die Durchführung grundlegender Textanalysen.
- **Kosten-Nutzen-Abwägung**: Die Kombination von GPT-4o und GPT-4o-mini ermöglicht eine optimale Balance zwischen Leistungsfähigkeit und Kosten. Während GPT-4o für die anspruchsvolleren Aufgaben eingesetzt wird, sorgt GPT-4o-mini dafür, dass weniger komplexe Aufgaben ressourcenschonend bearbeitet werden können. Diese strategische Nutzung der Modelle hilft, die Gesamtkosten des Projekts zu kontrollieren, ohne dabei die Qualität der Ergebnisse zu beeinträchtigen.

Die Entscheidung, sowohl GPT-4o als auch GPT-4o-mini zu verwenden, basiert auf einer sorgfältigen Abwägung zwischen der benötigten Leistungsfähigkeit und den verfügbaren Ressourcen. GPT-4o bietet die notwendige Präzision und Tiefe für komplexe Aufgaben, während GPT-4o-mini eine kosteneffiziente Lösung für weniger anspruchsvolle Aufgaben darstellt. Diese Kombination ermöglicht es, die Vorteile beider Modelle optimal zu nutzen und so die Effizienz und Effektivität des Projekts zu maximieren.

1.4 Quellcode und Prompts

Der Python-Quellcode für die Tools `analyse cmd.py` und `aictrl2.py`, sowie die verwendeten Prompts, sind auf GitHub verfügbar. Sie können diese unter folgendem Link einsehen und herunterladen:

GitHub Repository: `https://github.com/dirnberg/aictrl`

In diesem Repository finden Sie alle relevanten Dateien, die für das Projekt erstellt wurden, einschließlich der Konfigurationsdateien und Beispiel-Prompts. Das Repository bietet eine vollständige Übersicht über den Code und ermöglicht es Ihnen, die Tools nach Bedarf anzupassen oder weiterzuentwickeln.

2 Funktionsweise der Tools

2.1 Analyse von Texten mit `analyse cmd.py`

`analyse cmd.py` ist ein Python-basiertes Tool, das Texte aus verschiedenen Dokumentenformaten wie PDF, PowerPoint und HTML extrahiert und verarbeitet. Es bietet folgende Funktionen:

- **Textextraktion:** Extrahiert Inhalte aus PDF-Dokumenten, Präsentationen und Webseiten.
- **Übersetzung und Erklärung:** Automatische Übersetzung von Texten und Generierung von Erklärungen oder Zusammenfassungen.
- **Berichterstellung:** Erstellung strukturierter Berichte oder Markdown-Dateien basierend auf den extrahierten Inhalten.

Beispiel: Ein Unternehmen nutzt `analyse cmd.py`, um den Text aus einer Präsentation zu extrahieren, ins Englische zu übersetzen und eine leicht verständliche Zusammenfassung zu erstellen, die als Grundlage für eine Veröffentlichung dient.

2.2 Dokumentenverarbeitung und Analyse mit `aictrl2.py`

`aictrl2.py` analysiert die Texte, die von `analyse cmd.py` extrahiert wurden und auch weitere Dokumente, und generiert neue Inhalte, wie z.B. Berichte, Zusammenfassungen oder Empfehlungen. Das Tool bietet:

- **Textanalyse:** Prüfung auf Vollständigkeit, Konsistenz und potenzielle Risiken.
- **Verwendung von Platzhaltern:** Flexibilität durch Platzhalter in Prompts, die eine Anpassung an verschiedene Anwendungsfälle ermöglichen.
- **Erstellung neuer Inhalte:** Generierung von Berichten, die als eigenständige Dokumente oder zur Integration in ein Content-Management-System (CMS) genutzt werden können.

Beispiel: Ein Unternehmen setzt `aictrl2.py` zur Analyse eines Vertrags ein, um potenzielle Konflikte zu identifizieren und einen detaillierten Bericht mit Lösungsvorschlägen zu erstellen.

3 Technische Details und Konfigurationsmöglichkeiten

3.1 Konfigurationsdatei `config.yml`

Die Konfigurationsdatei `config.yml` steuert das Verhalten von `aictrl2.py` und ermöglicht eine flexible Anpassung an verschiedene Anforderungen. Hier ein Beispiel für eine typische Konfiguration:

```
assistant:
  name: "LegalDocumentAnalyzer"
  model: "gpt-4"
  instructions_file_path: "prompts/instructions.md"
  tools:
    - "file_search"
    - "code_interpreter"
  temperature: 0.7
  top_p: 0.9
  frequency_penalty: 0.0
  presence_penalty: 0.0
```

- **`name`**: Name des Assistenten.
- **`model`**: Das zu verwendende KI-Modell (z.B. GPT-4).
- **`instructions_file_path`**: Pfad zur Datei mit den Anweisungen für den Assistenten.
- **`tools`**: Eingesetzte Werkzeuge wie `file_search` und `code_interpreter`.
- **`temperature`**: Steuerung der Kreativität des Outputs.
- **`top_p`**: Parameter zur Probabilitätsverteilung.
- **`frequency_penalty`**: Bestrafung für häufige Wiederholungen.
- **`presence_penalty`**: Bestrafung für das Einführen neuer Konzepte.

3.2 Verwendung von Prompts und Platzhaltern**

Platzhalter in Prompts bieten eine hohe Flexibilität und Anpassungsfähigkeit. Sie ermöglichen es, denselben Prompt für verschiedene Szenarien zu verwenden, indem spezifische Parameter einfach ausgetauscht werden. Hier ein Beispiel für einen Prompt:

```
 m
You are a legal expert tasked with drafting an extremely

formal Legal Opinion Letter for a client in language: {{

language }} based on your knowledge base. Please review the

contents of the uploaded file(s) before answering.
```

```
Refer to the provided file(s) for information on {{
topic_or_law }}.
Use the details from the attached file(s) and, after examining
the uploaded document(s), create the specific work.
```

- `{{ language }}`: Platzhalter für die Sprache des Dokuments.
- `{{ topic_or_law }}`: Platzhalter für das spezifische Thema oder Gesetz.

4 Praktische Anwendungsbeispiele

4.1 Vertragsanalyse

Ein Unternehmen nutzt `aictrl2.py` zur Analyse eines Vertragsentwurfs. Das Tool prüft den Vertrag auf Vollständigkeit, Konsistenz und potenzielle Risiken und erstellt anschließend einen Bericht, der diese identifiziert und Lösungsvorschläge macht. Durch die Verwendung von Platzhaltern in den Prompts kann das Tool flexibel an verschiedene Vertragsarten angepasst werden.

Beispiel: Ein internationales Unternehmen analysiert einen Vertrag mit einem neuen Lieferanten und identifiziert mit `aictrl2.py` potenzielle Risiken in Bezug auf die Lieferbedingungen und Zahlungsfristen.

4.2 Projektdokumentation

Ein Projektteam setzt `analyse cmd.py` ein, um die gesamte Projektdokumentation zu prüfen. Das Tool extrahiert die Texte aus verschiedenen Dokumentenformaten, übersetzt sie bei Bedarf und erstellt eine konsolidierte Zusammenfassung der Projektdokumentation. `aictrl2.py` analysiert dann die Dokumentation und generiert Empfehlungen zur Verbesserung und Vervollständigung.

Beispiel: Ein Bauunternehmen verwendet die Tools, um sicherzustellen, dass alle Projektdokumentationen für ein Großprojekt vollständig und konsistent sind, bevor sie an den Auftraggeber übergeben werden.

4.3 Ausschreibungsprüfung

Bei der Teilnahme an einer öffentlichen Ausschreibung nutzt ein Unternehmen `aictrl2.py`, um die Ausschreibungsunterlagen zu prüfen. Das Tool analysiert die Anforderungen der

Ausschreibung, vergleicht sie mit den Unternehmensstandards und erstellt einen Bericht, der aufzeigt, ob das Unternehmen die Anforderungen erfüllt und welche Anpassungen notwendig sind.

Beispiel: Ein IT-Dienstleister analysiert eine Ausschreibung für ein großes Regierungsprojekt und stellt sicher, dass alle Anforderungen erfüllt sind, bevor das Angebot eingereicht wird.

5 Zukunftsperspektiven und Weiterentwicklungsmöglichkeiten

Das Projekt hat großes Potenzial für zukünftige Entwicklungen:

- **Integration zusätzlicher KI-Modelle:** Weitere KI-Modelle könnten integriert werden, um die

Analysefähigkeit des Tools zu erweitern.

- **On-Premise-Lösungen:** Durch den Einsatz auf eigener Hardware könnten Unternehmen das Tool lokal nutzen, um sensible Daten sicher zu verarbeiten.
- **Erweiterung auf neue Anwendungsbereiche:** Das Tool könnte für die Verarbeitung neuer Dokumentenformate oder in neuen Branchen eingesetzt werden, wie z.B. im Gesundheitswesen oder in der Finanzdienstleistung.
- **Multimodale Analysen:** Eine mögliche Erweiterung wäre die Fähigkeit, nicht nur Text, sondern auch Bilder, Videos oder Audioinhalte zu analysieren.
- **Einfache Benutzeroberfläche:** Die Entwicklung einer webbasierten Benutzeroberfläche könnte das Tool auch für nicht-technische Benutzer zugänglicher machen.

6 Herausforderungen und Lösungen

6.1 Konsistenz in Prompts

Eine der Herausforderungen bestand darin, die Anrede in den Prompts konsistent zu halten (Du/Sie). Es war notwendig, die Prompts nachträglich anzupassen, um sicherzustellen, dass die Ansprache einheitlich ist und keine Verwirrung verursacht.

6.2 Technische Schwierigkeiten bei der Python-Integration

Die Integration der OpenAI-Bibliotheken in Python brachte Herausforderungen mit sich, insbesondere die Kompatibilität verschiedener Bibliotheksversionen und die Notwendigkeit, den Code mehrfach zu testen. Eine sorgfältige Dokumentation und Versionierung der verwendeten Bibliotheken halfen, diese Schwierigkeiten zu überwinden.

6.3 Unvollständige Dokumentation der OpenAI-Bibliotheken

Die unvollständige oder nicht immer eindeutige Dokumentation der OpenAI-Bibliotheken machte es schwierig, die Funktionen optimal zu nutzen. Durch eine Kombination aus intensiver Recherche und iterativem Testen konnten diese Herausforderungen gemeistert werden.

7 Zusammenfassung und Fazit

Das Projekt `analyse cmd.py` und `aictrl2.py` bietet eine leistungsstarke Lösung für die automatisierte Textanalyse und Dokumentenverarbeitung. Durch die Kombination dieser Tools können Unternehmen komplexe Dokumentenverarbeitungsaufgaben effizient automatisieren und so Zeit sparen, Fehler minimieren und die Qualität der verarbeiteten Dokumente erheblich verbessern.

Die Flexibilität der Tools, unterstützt durch Platzhalter in den Prompts und umfangreiche Konfigurationsmöglichkeiten, ermöglicht es, sie in einer Vielzahl von Anwendungsbereichen einzusetzen. Die Zukunftsperspektiven für das Projekt sind vielversprechend, mit zahlreichen Erweiterungsmöglichkeiten, die das Potenzial der Tools weiter steigern können.