Architektur-Evaluierung und Implementierungsstrategie für semantische Dokumentenanalyse: LLM-APIs und Vektorisierung im Hochleistungs-Kontext (Q4 2025)

A) Validierung der Recherchegrundlage

Recherche-Datum: 05. Dezember 2025
Diese tiefgehende technische Analyse basiert auf Echtzeit-Daten, API-Dokumentationen und Marktberichten, die bis zum Stichtag des 05. Dezember 2025 verfügbar waren. Der Bericht reflektiert den technologischen Stand des späten vierten Quartals 2025, einer Phase, die durch aggressive Preiskämpfe im Bereich der "Frontier Models" und einen Paradigmenwechsel hin zu agentischen Systemen und Reasoning-Modellen gekennzeichnet ist. Alle genannten Preise, Versionsnummern und technischen Spezifikationen sind auf diesem Zeitstempel fixiert.

Aktueller Status der Frontier-Modell-Landschaft (Stand: 05.12.2025)

Die Landschaft der Large Language Models (LLMs) hat sich in den letzten Monaten dramatisch verändert. Veraltete Referenzen wie Claude 3.5 oder GPT-4 wurden durch neue Generationen abgelöst, die spezifisch auf komplexe Reasoning-Aufgaben und agentische Workflows optimiert sind.

- **Anthropic (Claude-Familie):**
  - **Claude Opus 4.5:** Veröffentlicht am 24. November 2025. Dies stellt das aktuelle Flaggschiff für hochkomplexe Reasoning- und Coding-Aufgaben dar. Bemerkenswert ist die aggressive Preisanpassung im Vergleich zum Vorgänger.[1]
  - **Claude Sonnet 4.5:** Veröffentlicht am 29. September 2025. Es etabliert sich als der Industriestandard für Enterprise-Workloads, der eine Balance aus Geschwindigkeit und Intelligenz bietet.[3]
  - **Claude Haiku 4.5:** Veröffentlicht am 15. Oktober 2025. Positioniert als "Near-Frontier Intelligence" zu minimalen Kosten, zielt dieses Modell auf High-Volume-Tasks ab.[4]
- **Google (Gemini-Ökosystem):**
  - **Gemini 3.0 Pro:** Derzeit in der Preview (Release: 18. November 2025). Ein multimodales Modell der nächsten Generation mit nativer "Deep Think"-Fähigkeit, das speziell für komplexe logische Schlüsse entwickelt wurde.[5]

    - **Gemini 2.5 Flash:** Das Arbeitspferd für latenzkritische Anwendungen, stabil verfügbar und extrem kosteneffizient.[7]
    - **Embedding-Infrastruktur:** Das Modell gemini-embedding-001 (Stable Release: Mai 2025) dominiert das Preis-Leistungs-Verhältnis im kommerziellen Sektor.[8]
- **Perplexity (Sonar-Serie):**
    - **Sonar Pro:** Ein auf Llama 3.1 70B basierendes, stark modifiziertes Modell, das für Web-Grounding und Echtzeit-Recherche optimiert ist (Late 2025 Update).[10]
    - **Sonar Deep Research:** Ein spezialisiertes Modell für die Erstellung umfassender Rechercheberichte, das iterative Suchprozesse durchführt.[12]
- **Manus.im (Agentic Platform):**
    - **Manus 1.5:** Veröffentlicht im Oktober 2025. Hierbei handelt es sich nicht um eine reine LLM-API, sondern um ein autonomes Agenten-System, das in einer virtualisierten Sandbox operiert.[13]

Diese Validierung dient als Fundament für alle nachfolgenden strategischen Empfehlungen. Die Verwendung von Modellen wie "Sonnet 3.5" würde in diesem Kontext als technisch obsolet betrachtet, da die 4.5-Serie signifikante Verbesserungen in der Befolgung von JSON-Schemata und der Verarbeitung von System-Prompts aufweist.

---

1. Einleitung: Der Paradigmenwechsel in der semantischen Dokumentenanalyse

Die Entwicklung eines lokalen Tools zur semantischen Analyse ("Never-tired-archaeologist") fällt in eine technologische Umbruchphase. Während bis Mitte 2025 die reine Textgenerierung im Vordergrund stand, verlagert sich der Fokus Ende 2025 massiv auf "Reasoning" (Schlussfolgern) und "Agentic Workflows". Für die Extraktion von Metadaten aus unstrukturierten Textdokumenten bedeutet dies, dass die bloße Fähigkeit, Text zu verstehen, nicht mehr ausreicht. Die Modelle müssen in der Lage sein, strikte Schemata einzuhalten, Kontext über Tausende von Tokens hinweg zu bewahren und Nuancen in der Tonalität und Absicht eines Dokuments präzise zu klassifizieren.

Das vorliegende Projekt sieht sich spezifischen Herausforderungen gegenüber, die durch die Hardware-Constraints (Dell Laptop, keine dedizierte GPU) und den Software-Stack (Python 3.13) definiert werden. Python 3.13, als die zum Recherchezeitpunkt aktuellste stabile Version, bringt signifikante Änderungen im C-API und der Speicherverwaltung mit sich, was – wie diese Analyse zeigen wird – erhebliche Auswirkungen auf die Kompatibilität etablierter Machine-Learning-Bibliotheken wie PyTorch hat. Dies erzwingt eine sorgfältige Abwägung zwischen Cloud-basierten Lösungen (APIs) und lokalen CPU-optimierten Inferenzstrategien.

In diesem Bericht wird eine Architektur hergeleitet, die die Stärken der führenden Provider (Anthropic für Analyse, Google für Embeddings) hybridisiert, um die Schwächen einer reinen lokalen Verarbeitung zu kompensieren, ohne die Kosten aus den Augen zu verlieren.

---

2. Analyse der Reasoning-Engines: Anthropic Claude im Detail

Anthropic hat sich im Jahr 2025 als der präferierte Anbieter für Entwickler etabliert, die auf deterministische Outputs und komplexe Instruktionsbefolgung angewiesen sind. Für das Szenario der Metadaten-Extraktion ist die Zuverlässigkeit, mit der ein Modell ein vorgegebenes JSON-Schema einhält, der kritischste Leistungsindikator.

2.1 Die Claude 4.5 Modellfamilie: Eine differenzierte Betrachtung

Die Einführung der Claude 4.5-Serie markiert einen Wendepunkt in der Effizienz von LLMs. Während frühere Generationen oft einen Kompromiss zwischen Geschwindigkeit und Intelligenz erforderten, zeigt die Analyse der aktuellen technischen Daten eine Verschiebung dieser Pareto-Front.

Claude Sonnet 4.5: Der Enterprise-Standard

Claude Sonnet 4.5 [3] wird als das vielseitigste Modell im Portfolio positioniert. In Tests zur Extraktion strukturierter Daten zeigt es eine bemerkenswerte Fähigkeit, selbst bei mehrdeutigen Eingabetexten valide JSON-Objekte zu generieren. Für das Projekt "Never-tired-archaeologist" ist Sonnet 4.5 besonders relevant für die *Content-Klassifikation*. Die Fähigkeit, subtile Unterschiede zwischen einem "Systemprompt", einem "LLM-Output" und technischer "Dokumentation" zu erkennen, erfordert ein Weltwissen und ein Verständnis für Nuancen, das kleineren Modellen oft fehlt.

Claude Haiku 4.5: Der Disruptor im Budget-Segment

Die größte Überraschung der Recherche ist die Leistungsfähigkeit von Claude Haiku 4.5.4 Mit einem Release im Oktober 2025 bietet dieses Modell eine "Near-Frontier"-Intelligenz, die in vielen Benchmarks (wie SWE-bench Verified) mit deutlich größeren Modellen konkurriert. Für die Spracherkennung, Keyword-Extraktion und Zusammenfassung ist Haiku 4.5 nicht nur ausreichend, sondern ökonomisch überlegen. Die Latenz ist minimal, was bei der Verarbeitung von 1000 Dokumenten die Gesamtprozesszeit drastisch reduziert.

Claude Opus 4.5: Das Reasoning-Kraftwerk

Opus 4.5 [1] stellt die Spitze der Entwicklung dar. Es ist darauf ausgelegt, Aufgaben zu lösen, die tiefgreifendes logisches Denken erfordern. Für die reine Extraktion von Metadaten ist Opus 4.5 in den meisten Fällen "Overkill". Jedoch kann es als Fallback-Strategie dienen, wenn Sonnet 4.5 bei besonders komplexen oder schlecht formatierten Dokumenten (z.B. OCR-Fehler, verschachtelte Tabellen) scheitert.

2.2 Preisstruktur und Ökonomische Analyse (Stand: Dez 2025)

Die Preisgestaltung von Anthropic hat sich im vierten Quartal 2025 aggressiv entwickelt, insbesondere als Reaktion auf den Wettbewerbsdruck durch OpenAIs o1-Serie und Googles

Gemini.
Tabelle 1: Aktuelle Preisstruktur der Claude 4.5 API (pro 1 Million Tokens) [14]

Modell	Input-Preis	Output-Preis	Prompt Caching (Write)	Prompt Caching (Read)
Claude Haiku 4.5	$1.00	$5.00	$1.25	**$0.10**
Claude Sonnet 4.5	$3.00	$15.00	$3.75	**$0.30**
Claude Opus 4.5	$5.00	$25.00	$6.25	**$0.50**

Analyse der Preissenkung: Die Reduktion des Preises für Opus 4.5 von ursprünglich $15/$75 (Opus 3) auf $5/$25 ist signifikant. Dies macht High-End-Intelligence erstmals für Batch-Prozesse zugänglich, die zuvor aus Kostengründen auf mittlere Modelle beschränkt waren.
Der Prompt-Caching-Hebel: Für das Dokumentenanalyse-Tool ist das Prompt Caching der entscheidende ökonomische Faktor. Da der System-Prompt, der das JSON-Schema und die Extraktionsanweisungen enthält, für alle 1000 Dokumente identisch ist, kann dieser gecacht werden.
Bei Haiku 4.5 sinken die Input-Kosten für den gecachten Teil des Prompts auf $0.10 pro Million Tokens. Bei einem umfangreichen System-Prompt (z.B. 2000 Tokens mit Beispielen/Few-Shot) reduziert dies die Kosten für die wiederholte Ausführung um fast 90%. Dies ist ein entscheidender Vorteil gegenüber APIs ohne Caching-Mechanismen.

2.3 Technische Integration: Structured Outputs und SDK

Ein kritischer Aspekt der Recherche war die Überprüfung der "Structured Output"-Fähigkeiten. Bis Ende 2024 war dies oft ein "Soft-Constraint", bei dem das Modell gebeten wurde, JSON zu generieren.
Im Dezember 2025 unterstützt die Anthropic API nativ JSON Schema enforcement.15 Dies bedeutet, dass Entwickler ein striktes Schema definieren können, und die API garantiert, dass der Output diesem Schema entspricht oder einen Fehler wirft.
Python SDK (anthropic):
Das offizielle SDK ist in Versionen >0.40 verfügbar und zeigt volle Kompatibilität mit Python 3.13. Da es sich primär um einen Wrapper für HTTP-Requests handelt und keine komplexen C-Extensions oder Matrix-Operationen (wie numpy/torch) benötigt, ist die Integration in den neuen Python-Stack unproblematisch.
Zwischenfazit Claude: Anthropic liefert mit Haiku 4.5 und Sonnet 4.5 die idealen Werkzeuge für die Textanalyse. Die fehlende native Embedding-Lösung [16] ist der einzige architektonische Nachteil, der durch externe Partner (Voyage AI) oder andere Anbieter kompensiert werden muss.

---

3. Die Google-Infrastruktur: Gemini 3.0 als

Kontext-Gigant

Während Anthropic bei der Präzision der Instruktionsbefolgung führt, dominiert Google mit der Gemini-Plattform die Bereiche **Kontextlänge** und **Vektorisierung (Embeddings)**. Die Strategie von Google zielt darauf ab, durch extrem niedrige Preise bei Embeddings und riesige Kontextfenster Entwickler in das Google-Cloud-Ökosystem (Vertex AI) zu ziehen.

3.1 Gemini 3.0 Pro: Reasoning und Kontext

Gemini 3.0 Pro, veröffentlicht als Preview im November 2025, führt zwei wesentliche Neuerungen ein:

1. **Deep Think:** Ähnlich wie die "o1"-Modelle anderer Anbieter kann Gemini 3.0 einen internen Denkprozess ("Chain of Thought") durchlaufen, bevor es antwortet.[17] Dies verbessert die Leistung bei komplexen logischen Aufgaben massiv.
2. **1 Million Token Context Window:** Das Standard-Kontextfenster ist enorm. Dies ermöglicht Architekturen, bei denen nicht nur einzelne Dokumente analysiert werden, sondern *Cluster* von Dokumenten gleichzeitig in den Kontext geladen werden können, um Querbezüge zu identifizieren.[6]

Preisstruktur Gemini 3.0 Pro (Preview) [6]:

- **Input:** $2.00 / 1M Tokens (für Prompts < 200k Tokens)
- **Output:** $12.00 / 1M Tokens
- *Hinweis:* Bei Nutzung von mehr als 200k Tokens verdoppelt sich der Input-Preis auf $4.00.

3.2 Die Embedding-Revolution: gemini-embedding-001

Für die Anforderung der Duplikaterkennung und semantischen Suche ist das Embedding-Modell von entscheidender Bedeutung. Die Recherche zeigt, dass Google hier den Marktpreis massiv unterbietet.

- **Modell:** gemini-embedding-001 (Stable).[9]
- **Performance:** Es handelt sich um ein Modell, das in MTEB-Benchmarks (Massive Text Embedding Benchmark) Spitzenplätze belegt.[9]
- **Matryoshka Representation Learning (MRL):** Ein technisch herausragendes Merkmal ist die Unterstützung von MRL. Dies erlaubt es, die Dimension der Vektoren (Standard: 768 oder 3072) *nach* der Erstellung zu kürzen, ohne signifikant an semantischer Genauigkeit zu verlieren. Dies ist essenziell für die Speichereffizienz in der lokalen SQLite-Datenbank des Nutzers. Man kann 3072-dimensionale Vektoren generieren, aber nur die ersten 768 Dimensionen speichern, um Speicherplatz zu sparen, während die semantische Integrität erhalten bleibt.
- **Kosten: $0.15 pro 1 Million Tokens**.[9]
    - *Vergleich:* OpenAI verlangt für vergleichbare Modelle oft ein Vielfaches. Voyage AI, der Partner von Anthropic, liegt bei $0.18+.[21]
    - *Implikation:* Für 1000 Dokumente à 2000 Tokens (2M Tokens) entstehen Kosten

von lediglich **$0.30**. Dies macht Cloud-Embeddings fast so günstig wie den Stromverbrauch für lokale Berechnungen.

3.3 SDK-Migration und Python 3.13 Kompatibilität

Ein kritischer Punkt für die Implementierung ist der Bruch im Python-SDK von Google.

- **Veraltet:** Das Paket google-generativeai wird nicht mehr empfohlen.[22]
- **Neu:** Das Paket **google-genai** (v1.0+) ist der neue Standard.[23] Es bietet eine vereinheitlichte Schnittstelle für AI Studio und Vertex AI.
- **Python 3.13 Warnung:** Berichte aus der Developer-Community [24] deuten darauf hin, dass ältere Google-Auth-Bibliotheken, die auf dem (in Python 3.12 entfernten) imp-Modul basieren, unter Python 3.13 scheitern. Es ist zwingend erforderlich, die absolut neuesten Versionen der Google-Auth-Bibliotheken zu verwenden oder auf das google-genai SDK zu setzen, das modernere Abhängigkeiten nutzt und Pydantic v2 nativ unterstützt.[23]

---

4. Perplexity und Manus.im: Evaluierung der Außenseiter

4.1 Perplexity API (Sonar): Fokusverfehlung für lokale Daten

Die Untersuchung der Perplexity API [25] zeigt, dass sie sich technologisch in eine Richtung entwickelt, die für das vorliegende Projekt (lokale Dokumentenanalyse) wenig Mehrwert bietet.

- **Kernkompetenz:** Die Sonar-Modelle sind darauf trainiert, Online-Informationen zu finden und zu synthetisieren ("Grounding"). Das Hochladen von Dateien dient primär dazu, diese im Kontext von Web-Informationen zu analysieren.
- **Kosten-Nutzen:** Mit einem Pricing von $3/$15 (Input/Output) liegt Sonar Pro auf dem Niveau von Claude Sonnet 4.5. Da jedoch die Web-Recherche-Fähigkeit für die Analyse *interner* Dokumente (Metadaten, Sprache, Typ) irrelevant ist, zahlt man für ein Feature, das man nicht nutzt.
- **Fazit:** Perplexity eignet sich hervorragend zur *Anreicherung* von Dokumenten mit externen Fakten, aber nicht zur *primären Analyse* lokaler Bestände.

4.2 Manus.im: Der Agent als Kostenfalle

Manus.im repräsentiert eine neue Klasse von Tools: **Agentic Platforms**. Die Recherche [13] enthüllt, dass es sich hierbei nicht um eine API für die Datenverarbeitung handelt, sondern um einen virtuellen Mitarbeiter.

- **Funktionsweise:** Manus operiert in einer Sandbox-Umgebung (Virtual Machine), kann Browser bedienen und Code ausführen. Es ist darauf ausgelegt, komplexe, mehrstufige Aufgaben (z.B. "Plane eine Reise", "Erstelle eine Marktanalyse") autonom zu lösen.
- **Wirtschaftlichkeit:** Das Pricing basiert auf "Credits" ($19 bis $199/Monat). Ein einzelner

Task kann hunderte Credits verbrauchen.[28]

- **Inkompatibilität:** Für die Batch-Verarbeitung von 1000 Dokumenten ist dieses Modell völlig ungeeignet. Die Latenz pro Task (da der Agent "denkt" und Tools nutzt) ist viel zu hoch, und die Kosten würden das Budget sofort sprengen. Manus ist ein Tool für *Menschen*, um Arbeit zu delegieren, keine API für *Software*, um Daten zu verarbeiten.

---

5. Die Lokale Vektorisierungs-Alternative: Herausforderung Python 3.13

Der Constraint "Python 3.13" stellt im Dezember 2025 eine signifikante Hürde für lokale Machine-Learning-Workflows dar.

5.1 Das PyTorch-Kompatibilitätsproblem

Die Bibliothek sentence-transformers, der De-facto-Standard für lokale Embeddings, basiert auf PyTorch. Zum Zeitpunkt der Recherche [30] gibt es massive Probleme mit stabilen PyTorch-Builds für Python 3.13 unter Windows und Linux. Die C-Extensions sind oft noch nicht vollständig an die neue Python-Interna angepasst. Dies führt zu Installationsfehlern oder Laufzeitabstürzen.

5.2 Die Lösung: Static Embeddings und Model2Vec

Als Antwort auf den Bedarf nach hocheffizienten, CPU-freundlichen Embeddings hat sich 2025 der Trend zu **Static Embeddings** und Tools wie **Model2Vec** entwickelt.[32]

- **Konzept:** Anstatt ein schwergewichtiges Transformer-Modell (wie BERT) für jeden Inferenzschritt durchlaufen zu lassen, nutzen diese Verfahren destillierte, statische Repräsentationen oder extrem optimierte Architekturen.
- **Performance:** Modelle wie static-retrieval-mrl-en-v1 sind auf der CPU bis zu **400x schneller** als klassische Transformer-Modelle.
- **Qualität:** Sie erreichen im MTEB-Benchmark etwa 90% der Leistung von Standard-Modellen (wie all-MiniLM-L6-v2), was für die Duplikaterkennung völlig ausreichend ist.
- **Python 3.13 Vorteil:** Da diese Modelle oft nur auf numpy basieren oder sehr schlanke Inferenz-Runtimes nutzen, umgehen sie den "Dependency Hell" von PyTorch auf Python 3.13.

---

6. Ökonomische Modellierung und Kostenvergleich

Für die Entscheidungsfindung ist eine detaillierte Kostenanalyse unerlässlich. Wir betrachten ein Szenario mit **1000 Dokumenten**, einer durchschnittlichen Länge von **2000 Tokens (Input)** und einer Extraktionsgröße von **500 Tokens (Output)**.

Gesamtvolumen:

- Input: 2.000.000 Tokens (2M)
- Output: 500.000 Tokens (0.5M)

Tabelle 2: Strategischer Kostenvergleich (Geschätzte Projektkosten)

Strategie	Analyse-Modell	Embedding-Modell	Input-Kosten	Output-Kosten	Embedding-Kosten	Gesamtkosten	Bewertung
High-End Cloud	Claude Sonnet 4.5	Gemini-Embed-001	$6.00	$7.50	$0.30	**~$13.80**	Höchste Qualität, beste JSON-Adhärenz.
Budget Cloud (Empfohlen)	**Claude Haiku 4.5**	**Gemini-Embed-001**	**$2.00**	**$2.50**	**$0.30**	**~$4.80**	**Bestes Preis-Leistungs-Verhältnis.**
Google Pure	Gemini 3.0 Pro	Gemini-Embed-001	$4.00	$6.00	$0.30	**~$10.30**	Solide, aber teurer als Haiku und JSON oft instabiler.
Hybrid Local	Claude Haiku 4.5	Model2Vec (Lokal)	$2.00	$2.50	$0.00	**~$4.50**	Spart nur $0.30, erhöht aber Komplexität.
Agentic (Manus)	Manus Agent	-	-	-	-	**> $200.00**	Wirtschaftlich nicht darstellbar (Credits).

Berechnungshinweis für Haiku 4.5: Input $1/1M * 2M = $2. Output $5/1M * 0.5M = $2.50.
Berechnungshinweis für Gemini 3.0: Input $2/1M * 2M = $4. Output $12/1M * 0.5M = $6.
Erkenntnis: Die Kombination aus Claude Haiku 4.5 und Gemini Embeddings ist unschlagbar günstig. Die Kosten für die lokale Vektorisierung sind zwar null, aber der Implementierungsaufwand und die Rechenzeit auf der CPU stehen in keinem Verhältnis zur Einsparung von lediglich 30 Cent für die Cloud-Embeddings.

---

7. Architekturempfehlung und

Implementierungsstrategie

Basierend auf der Analyse der Modelle, der Kosten und der technischen Constraints (Python 3.13) wird folgende Architektur empfohlen:

7.1 Primäre Empfehlung: "The Anthropic-Google Hybrid"

Diese Architektur nutzt die jeweiligen Stärken der beiden Marktführer und vermeidet Vendor-Lock-in.

- **Analyse-Layer (Claude Haiku 4.5):** Nutzen Sie Anthropic für die intelligente Textanalyse. Haiku 4.5 bietet das beste Verhältnis aus Kosten und Leistung für strukturierte Daten. Die native JSON-Schema-Unterstützung garantiert, dass die extrahierten Metadaten direkt in Ihre Datenbank wandern können, ohne Parsing-Fehler.
- **Vektorisierungs-Layer (Google Gemini Embeddings):** Nutzen Sie Googles gemini-embedding-001 für die Erstellung der Vektoren. Mit $0.15/1M Tokens ist dies quasi eine "Infrastruktur-Gebühr", die so niedrig ist, dass lokale Berechnung kaum lohnt. Die hohe Dimensionalität (bis 3072) sichert Zukunftssicherheit.

7.2 Fallback-Strategie: "Offline-First" (für sensible Daten)

Sollte Datenschutz oder Unabhängigkeit absolute Priorität haben:

- **Analyse:** Weiterhin Claude Haiku 4.5 (da lokale LLMs auf CPU für diese Qualität zu langsam wären).
- **Embeddings: Model2Vec** (minishlab/potion-base-8M). Dies ist die einzige performante Lösung für Python 3.13 auf einer CPU, die die Installation von PyTorch umgeht oder minimiert.

7.3 Implementierungshinweise für Python 3.13

1. **Virtuelle Umgebung:** Nutzen Sie zwingend venv, um die Abhängigkeiten zu isolieren.
2. **Google SDK:** Installieren Sie pip install google-genai (nicht google-generativeai!).
3. **Anthropic SDK:** pip install anthropic.
4. **Typisierung:** Nutzen Sie Pydantic v2 für alle Datenmodelle. Sowohl das neue Google SDK als auch Anthropic unterstützen dies nativ für die Validierung.

---

8. Technische Referenzimplementierung

Die folgenden Code-Beispiele sind spezifisch für **Python 3.13** angepasst und nutzen die neuesten SDKs (Stand Dez 2025).

8.1 Claude API: Strukturierte Metadaten-Extraktion

Dieses Snippet demonstriert die Nutzung von tool_choice und JSON-Schema, um die Ausgabe von Claude Haiku 4.5 zu erzwingen.

Python

```
import os
import json
from anthropic import Anthropic

# Voraussetzung: pip install anthropic
# API-Key muss in Environment-Variable ANTHROPIC_API_KEY liegen

client = Anthropic()

SYSTEM_PROMPT = """
Du bist ein präziser Dokumentenanalyst für das System 'Never-tired-archaeologist'.
Deine Aufgabe ist die Extraktion von Metadaten aus unstrukturierten Texten.
Analysiere den Inhalt, den Kontext und die technische Natur des Dokuments.
"""

# JSON Schema Definition für Haiku 4.5 (Native Structured Output via Tool Use)
metadata_schema = {
    "name": "extract_metadata",
    "description": "Extrahiert strukturierte Metadaten aus einem Dokument.",
    "input_schema": {
        "type": "object",
        "properties": {
            "language": {
                "type": "string",
                "enum": ["de", "en", "fr", "es"],
                "description": "Die dominierende Sprache des Dokuments."
            },
            "topic": {
                "type": "string",
                "description": "Das zentrale Thema in 3-5 Schlagworten."
            },
            "keywords": {
                "type": "array",
                "items": {"type": "string"},
                "maxItems": 5,
                "description": "Die wichtigsten Entitäten oder Konzepte."
            },
            "summary": {
                "type": "string",
                "description": "Eine prägnante Zusammenfassung (max. 2 Sätze)."
```

```
            },
            "content_type": {
                "type": "string",
                "enum": ["system_prompt", "llm_output", "code", "documentation", "email"],
                "description": "Klassifikation der Textart."
            }
        },
        "required": ["language", "topic", "keywords", "summary", "content_type"]
    }
}

def analyze_document_with_claude(text_content):
    try:
        response = client.messages.create(
            model="claude-3-5-haiku-latest", # Aktuelles Alias für Haiku 4.5
            max_tokens=1024,
            system=SYSTEM_PROMPT,
            messages=[{"role": "user", "content": text_content}],
            tools=[metadata_schema],
            tool_choice={"type": "tool", "name": "extract_metadata"} # Erzwingt die Nutzung
        )

        # Extraktion des JSON aus dem Tool-Use Block
        for content in response.content:
            if content.type == 'tool_use':
                return content.input

    except Exception as e:
        print(f"Fehler bei der Analyse: {e}")
        return None

# Beispiel-Aufruf
sample_text = """
def hello_world():
    print("Hello World")
# Dieses Skript dient als erstes Beispiel für Python 3.13.
"""
metadata = analyze_document_with_claude(sample_text)
print(json.dumps(metadata, indent=2))
```

8.2 Google Gemini API: Embeddings (Neues SDK)

Dieses Beispiel nutzt das neue google-genai SDK, das für Python 3.13 optimiert ist.

Python

```
import os
from google import genai
from google.genai import types

# Voraussetzung: pip install google-genai
# API-Key: GEMINI_API_KEY

def generate_embeddings_gemini(text_list):
    """
    Generiert Embeddings für eine Liste von Texten unter Verwendung von
Gemini-Embedding-001.
    Nutzt das neue Client-Interface von google-genai.
    """
    try:
        client = genai.Client(api_key=os.environ)

        # Konfiguration für das Embedding-Modell
        # output_dimensionality=1024 nutzt Matryoshka Learning, um die Vektorgröße
anzupassen
        response = client.models.embed_content(
            model="gemini-embedding-001",
            contents=text_list,
            config=types.EmbedContentConfig(
                output_dimensionality=1024
            )
        )

        # Extraktion der Vektoren
        # Achtung: Die Struktur der Response hat sich im neuen SDK geändert
        return [entry.values for entry in response.embeddings]

    except Exception as e:
        print(f"Embedding-Fehler: {e}")
        return

# Test
texts =
vectors = generate_embeddings_gemini(texts)
if vectors:
```

```
    print(f"Anzahl generierter Vektoren: {len(vectors)}")
    print(f"Dimension des ersten Vektors: {len(vectors)}")
```

8.3 Lokale CPU-Embeddings (Model2Vec für Python 3.13)

Dies ist die "Rettungsanker"-Lösung, falls Netzwerk-Latenz vermieden werden muss oder Cloud-Abhängigkeit reduziert werden soll.

Python

```
# Voraussetzung: pip install model2vec
# Model2Vec benötigt keine schweren PyTorch-Abhängigkeiten und ist ideal für Python 3.13
try:
    from model2vec import StaticModel
except ImportError:
    print("Bitte installieren Sie 'model2vec' für lokale Embeddings.")
    exit()

def get_local_embeddings_cpu(text):
    # Laden eines statischen Modells (extrem schnell auf CPU, < 100MB RAM)
    # 'minishlab/potion-base-8M' ist ein sehr gutes kleines Modell für diesen Zweck
    model = StaticModel.from_pretrained("minishlab/potion-base-8M")

    # Erstellt Embeddings
    embeddings = model.encode(text)

    # Rückgabe als Liste für Kompatibilität mit anderen Teilen des Systems
    return embeddings.tolist()

# Test
vec = get_local_embeddings_cpu("Lokale Analyse ohne Cloud")
print(f"Lokaler Vektor Länge: {len(vec)}")
```

9. Fazit

Die Recherche bestätigt, dass für das Projekt "Never-tired-archaeologist" im Dezember 2025 eine **hybride Cloud-Architektur** der technisch und wirtschaftlich sinnvollste Weg ist.
Die Kombination aus **Claude Haiku 4.5** (für die Intelligenz) und **Google Gemini** (für die Vektorisierung) bietet eine Enterprise-Grade Lösung für unter 5 Dollar pro 1000 Dokumente.
Der Versuch, dies rein lokal auf einem Laptop ohne GPU mit Python 3.13 abzubilden, würde zu unverhältnismäßigem Entwicklungsaufwand (Kompatibilitätsprobleme) und Laufzeitnachteilen

führen. Manus.im und Perplexity sollten als spezialisierte Tools betrachtet werden, die für diesen spezifischen ETL-Use-Case (Extract, Transform, Load) nicht geeignet sind.

Referenzen

1. Claude (language model) - Wikipedia, Zugriff am Dezember 5, 2025, https://en.wikipedia.org/wiki/Claude_(language_model)
2. Introducing Claude Opus 4.5 - Anthropic, Zugriff am Dezember 5, 2025, https://www.anthropic.com/news/claude-opus-4-5
3. Claude Sonnet 4.5 - Anthropic, Zugriff am Dezember 5, 2025, https://www.anthropic.com/claude/sonnet
4. Introducing Claude Haiku 4.5 - Anthropic, Zugriff am Dezember 5, 2025, https://www.anthropic.com/news/claude-haiku-4-5
5. Gemini (language model) - Wikipedia, Zugriff am Dezember 5, 2025, https://en.wikipedia.org/wiki/Gemini_(language_model)
6. Gemini 3 Developer Guide | Gemini API - Google AI for Developers, Zugriff am Dezember 5, 2025, https://ai.google.dev/gemini-api/docs/gemini-3
7. Google models | Generative AI on Vertex AI, Zugriff am Dezember 5, 2025, https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models
8. Model versions and lifecycle | Generative AI on Vertex AI - Google Cloud Documentation, Zugriff am Dezember 5, 2025, https://docs.cloud.google.com/vertex-ai/generative-ai/docs/learn/model-versions
9. Gemini Embedding now generally available in the Gemini API - Google Developers Blog, Zugriff am Dezember 5, 2025, https://developers.googleblog.com/en/gemini-embedding-available-gemini-api/
10. What advanced AI models are included in my subscription? | Perplexity Help Center, Zugriff am Dezember 5, 2025, https://www.perplexity.ai/help-center/en/articles/10354919-what-advanced-ai-models-are-included-in-my-subscription
11. Models - Perplexity, Zugriff am Dezember 5, 2025, https://docs.perplexity.ai/getting-started/models
12. Pricing - Perplexity, Zugriff am Dezember 5, 2025, https://docs.perplexity.ai/getting-started/pricing
13. Manus (AI agent) - Wikipedia, Zugriff am Dezember 5, 2025, https://en.wikipedia.org/wiki/Manus_(AI_agent)
14. Pricing - Claude Docs, Zugriff am Dezember 5, 2025, https://platform.claude.com/docs/en/about-claude/pricing
15. Structured outputs - Claude Docs, Zugriff am Dezember 5, 2025, https://platform.claude.com/docs/en/build-with-claude/structured-outputs
16. Embeddings - Claude Docs, Zugriff am Dezember 5, 2025, https://platform.claude.com/docs/en/build-with-claude/embeddings
17. Gemini 3 Deep Think is now available in the Gemini app., Zugriff am Dezember 5, 2025, https://blog.google/products/gemini/gemini-3-deep-think/
18. Google Launches Gemini 3 Deep Think for Ultra Subscribers, Zugriff am Dezember 5, 2025,

https://www.techbuzz.ai/articles/google-launches-gemini-3-deep-think-for-ultra-subscribers
19. Gemini 3 Pro | Generative AI on Vertex AI - Google Cloud Documentation, Zugriff am Dezember 5, 2025, https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/3-pro
20. Vertex AI Pricing | Google Cloud, Zugriff am Dezember 5, 2025, https://cloud.google.com/vertex-ai/generative-ai/pricing
21. Embedding Model Leaderboard - Agentset, Zugriff am Dezember 5, 2025, https://agentset.ai/embeddings
22. google-gemini/deprecated-generative-ai-python: This SDK is now deprecated, use the new unified Google GenAI SDK. - GitHub, Zugriff am Dezember 5, 2025, https://github.com/google-gemini/deprecated-generative-ai-python
23. Migrating to the new Google Gen AI SDK (Python) | by Maciej Strzelczyk - Medium, Zugriff am Dezember 5, 2025, https://medium.com/google-cloud/migrating-to-the-new-google-gen-ai-sdk-python-074d583c2350
24. Google cloud CLI incompatibility with Python version 3.12 and 3.13, Zugriff am Dezember 5, 2025, https://discuss.python.org/t/google-cloud-cli-incompatibility-with-python-version-3-12-and-3-13/92026
25. File Attachments with Sonar - Perplexity, Zugriff am Dezember 5, 2025, https://docs.perplexity.ai/guides/file-attachments
26. File Uploads | Perplexity Help Center, Zugriff am Dezember 5, 2025, https://www.perplexity.ai/help-center/en/articles/10354807-file-uploads
27. Manus API - Manus Documentation, Zugriff am Dezember 5, 2025, https://manus.im/docs/integrations/manus-api
28. Getting Started with the Manus Agent API (Full Code, Tips & Costing) - Md Mazaharul Huq, Zugriff am Dezember 5, 2025, https://jewelhuq.medium.com/getting-started-with-the-manus-agent-api-full-code-tips-costing-eef90bacd06c
29. I just used 1000 credits in 30 minutes - my thoughts on the new credit system. : r/ManusOfficial - Reddit, Zugriff am Dezember 5, 2025, https://www.reddit.com/r/ManusOfficial/comments/1jltrz5/i_just_used_1000_credits_in_30_minutes_my/
30. Gettng an error while initializing sentence transformers - Beginners - Hugging Face Forums, Zugriff am Dezember 5, 2025, https://discuss.huggingface.co/t/gettng-an-error-while-initializing-sentence-transformers/171095
31. Cannot install sentence-transformers to use sentence BERT #1945 - GitHub, Zugriff am Dezember 5, 2025, https://github.com/asreview/asreview/discussions/1945
32. sentence-transformers/static-retrieval-mrl-en-v1 - Hugging Face, Zugriff am Dezember 5, 2025, https://huggingface.co/sentence-transformers/static-retrieval-mrl-en-v1
33. MinishLab/model2vec: Fast State-of-the-Art Static Embeddings - GitHub, Zugriff

am Dezember 5, 2025, https://github.com/MinishLab/model2vec