M17 - Modellauswahl_Evaluation

Stand: 04.2025

1 | KI-Modelle

Überblick der Modelltypen

Die moderne KI-Landschaft bietet verschiedene spezialisierte Modelltypen, die jeweils für bestimmte Anwendungsfälle optimiert sind:

- **Reasoning-Modelle**: Spezialisiert auf logisches Denken und Problemlösung (z.B. o3-mini)
- **Sprachmodelle**: Für natürlichsprachliche Aufgaben wie Textgenerierung und -verständnis (z.B. GPT-4)
- **Codex-Modelle**: Unterstützen bei der Codegenerierung und Programmieraufgaben
- **Bildgenerierungsmodelle**: Erzeugen kreative Bilder aus textlichen Beschreibungen (z.B. DALL-E)
- **Sprachverarbeitungsmodelle**: Für Spracherkennung und -transkription (z.B. Whisper)

Verständnis der Modellfunktionen

Jedes Modell hat spezifische Stärken und Schwächen. Die Auswahl des richtigen Modells für eine bestimmte Aufgabe erfordert ein Verständnis dieser Eigenschaften sowie der zugrundeliegenden Technologien.

2 | Modellübersicht

Tier	Modelle	Eigenschaften	Anwendungsfälle
Tier 3	- **GPT-4o mini** - Gemini 1.5 Flash - Claude 3 Haiku - LLaMA 3 - DeepSeek R1 - **o3-mini**	- Preiswert - Schnell - Ressourceneffizient - Hohe Intelligenz für die Größe	- Zusammenfassung vieler Dateien - Analyse von Podcast-Transkripten - Schnelle Textgenerierung - Lokale Verarbeitung auf Geräten - Prototyping und Experimente
Tier 2	- GPT-4 Turbo - Claude 3.7 Sonnet - Gemini 2.0 Pro - Qwen 2.5 Max - Microsoft Copilot	- Gutes Preis-Leistungs-Verhältnis - Vielseitig einsetzbar - Programmierung - Funktionsaufrufe - Tool-Nutzung	- E-Mail-Bearbeitung
Tier 1	- **GPT-4o** - Claude 3.5 Opus - Gemini 1.0 Ultra - DeepSeek R1 (High-End) - Mistral	- Höchste Intelligenz - Fortschrittliche Reasoning-Fähigkeiten - Langsamer und teurer - Tiefes Verständnis - Kontextbezogenes Denken	- Generierung von Anwendungsfällen - Komplexe Reasoning-Aufgaben

Tier 3: Die Arbeitspferde
Tier-3-Modelle sind kostengünstige, schnelle und ressourceneffiziente Optionen, ideal für Aufgaben mit hohen Geschwindigkeitsanforderungen und geringen Kosten.

Vorteile: Preiswert, schnell, ressourceneffizient
Anwendungen: Datenzusammenfassungen, Transkriptanalysen, schnelle Textgenerierung, lokale Verarbeitung
Beispiele: GPT-4o mini, Gemini 1.5 Flash, DeepSeek R1

Tier 2: Die Allrounder
Tier-2-Modelle bieten ein gutes Gleichgewicht zwischen Leistung und Kosten und eignen sich für vielseitige Anwendungen.

Vorteile: Vielseitig einsetzbar, gutes Preis-Leistungs-Verhältnis
Anwendungen: Programmierung, Funktionsaufrufe, Tool-Nutzung
Beispiele: GPT-4 Turbo, Claude 3.7 Sonnet, Qwen 2.5 Max

Tier 1: Die Spitzenreiter
Tier-1-Modelle repräsentieren die Spitze der KI-Technologie und sind für komplexe Aufgaben mit tiefem Verständnis konzipiert.

Vorteile: Höchste Intelligenz, fortschrittliche Reasoning-Fähigkeiten
Anwendungen: Kontextbezogenes Denken, Generierung von Anwendungsfällen
Beispiele: GPT-4o, Claude 3.5 Opus, Mistral

3 | Bewertung von Modellen

Quantitative Bewertungsansätze

Quantitative Ansätze liefern messbare Werte und ermöglichen objektive Vergleiche zwischen verschiedenen Modellen:

- **Perplexity**: Misst, wie gut das Modell Wahrscheinlichkeiten für das nächste Token vorhersagen kann (niedrigere Werte = besser)
- **Automatisierte Metriken**: BLEU, ROUGE oder METEOR zur Bewertung von Übersetzung oder Zusammenfassung
- **Benchmark-Tests**: Standardisierte Tests wie GLUE, SuperGLUE, SQuAD für spezifische Aufgaben

Qualitative Bewertungsansätze

Qualitative Ansätze erfassen Aspekte, die durch automatisierte Metriken oft nicht abgebildet werden können:

- **Human Evaluation**: Bewertung durch menschliche Experten hinsichtlich Kohärenz, Relevanz und Kreativität
- **Interaktive Evaluation**: Bewertung im praktischen Einsatz durch Nutzerfeedback in realen Anwendungsszenarien

Robustheit und Sicherheit

Zusätzliche wichtige Kriterien, um die Vertrauenswürdigkeit und ethische Einsetzbarkeit eines Modells zu bewerten:

- Reaktion auf unerwartete oder fehlerhafte Eingaben
- Anfälligkeit für Manipulation
- Vorhandensein von Bias oder Halluzinationen

4 | Modellauswahlprozess

Anforderungsanalyse

Der erste Schritt ist die genaue Definition der Anforderungen an das Modell:

- Welche Aufgaben soll das Modell übernehmen? (Textgenerierung, Fragen beantworten, Übersetzungen, etc.)
- Welche Qualitätskriterien sind wichtig? (Kohärenz, Genauigkeit, etc.)
- Welche Domänenkenntnisse werden benötigt?

Definition der Bewertungskriterien

Festlegung relevanter Kriterien für den Vergleich der Modelle:

- Verständlichkeit der Ausgaben
- Effizienz und Geschwindigkeit
- Skalierbarkeit
- Kosten

Recherche und Vorauswahl

Analyse bestehender Modelle (z.B. GPT-4, Claude, Gemini) anhand der festgelegten Kriterien.

Praktische Modellbewertung

Test der vielversprechendsten Modelle in einer realitätsnahen Umgebung:

- Quantitative Methoden (Benchmarking, Metriken)
- Qualitative Verfahren (Nutzerrückmeldungen)
- Testphase zur praktischen Erprobung

Finale Auswahl und Implementierung

Entscheidung für das am besten geeignete Modell und Integration in die bestehende Umgebung mit kontinuierlichem Monitoring.

5 | Modellkaskade

Die verschiedenen ChatGPT-Modelle können in einer Modellkaskade eingesetzt werden, um komplexe Aufgaben effizienter zu lösen. Dabei wird jedes Modell entsprechend seiner Stärken für spezifische Teilaufgaben genutzt. Hier ist ein Beispiel für eine solche Kaskade:

Erstellung eines wissenschaftlichen Berichts

Ziel: Automatisierte Erstellung eines Berichts, der Datenanalyse, Textgenerierung und kreative Präsentation kombiniert.

Schritt 1: Datenanalyse mit pandas

- **Einsatz:** Die Bibliothek wird verwendet, um große Datensätze zu analysieren und statistische Zusammenfassungen zu erstellen.
- **Vorteile:** Kostenlos und effizient bei MINT-Aufgaben.
- **Nachteil:** Begrenzte Fähigkeit bei komplexen Textanalysen.
- **Beispiel:** Analyse von Umfragedaten und Berechnung von Korrelationen.

Schritt 2: Logische Strukturierung mit ChatGPT o3-mini

- **Einsatz:** Das Modell strukturiert die Ergebnisse der Datenanalyse und erstellt eine logische Gliederung des Berichts.
- **Vorteile:** Hohe Präzision und Fähigkeit zur tiefgehenden Analyse.
- **Nachteil:** Höherer Ressourcenverbrauch.
- **Beispiel:** Formulierung der Hauptthesen und Ableitung von Schlussfolgerungen.

Schritt 3: Kreative Textgenerierung mit ChatGPT-4o

- **Einsatz:** Dieses Modell wird genutzt, um den Bericht in ansprechender Sprache zu verfassen und kreative Elemente wie Infografiken oder Storytelling einzubinden.
- **Vorteile:** Natürliches Sprachverhalten und kreative Fähigkeiten.
- **Nachteil:** Teuer und nur im Pro-Abo verfügbar.
- **Beispiel:** Erstellung des finalen Berichts mit einer überzeugenden Einleitung und klaren Argumentationslinien.

Schritt 4: Multimodale Präsentation mit gpt-4.5 und plotly-express

- **Einsatz:** Das Modell ergänzt den Bericht durch visuelle Elemente wie Diagramme oder Bilder und bereitet ihn für die Präsentation vor.
- **Vorteile:** Multimodale Fähigkeiten zur Verarbeitung von Text, Bild und Audio.
- **Nachteil:** Begrenzte Kapazität bei Spitzenzeiten.

- **Beispiel:** Generierung von Diagrammen basierend auf den analysierten Daten.

Code-Beispiel Modelkaskade

```
import pandas as pd
from openai import OpenAI
import plotly.express as px

# OpenAI-Client initialisieren
client = OpenAI()

# Schritt 1: Datenanalyse mit pandas
def daten_analyse(datei_pfad):
    """
    Führt eine grundlegende Datenanalyse durch, einschließlich statistischer
Zusammenfassung
    und Korrelationsmatrix.
    """
    # Beispiel-Datensatz laden
    daten = pd.read_csv(datei_pfad)

    # Statistische Analyse durchführen
    zusammenfassung = daten.describe()
    korrelationen = daten.corr()

    print("Statistische Zusammenfassung:")
    print(zusammenfassung)
    print("\nKorrelationsmatrix:")
    print(korrelationen)

    return daten, zusammenfassung, korrelationen

# Schritt 2: Logische Strukturierung mit o3-mini
def logische_strukturierung(zusammenfassung, korrelationen):
    """
    Erstellt eine logische Gliederung für einen wissenschaftlichen Bericht
basierend auf den
    Analyseergebnissen.
    """
    # Prompt für die Gliederung
    prompt = f"""
    Erstelle eine logische Gliederung für einen wissenschaftlichen Bericht
basierend auf den folgenden Daten:

    Statistische Zusammenfassung:
```

```
    {zusammenfassung}

    Korrelationsmatrix:
    {korrelationen}

    Die Gliederung sollte folgende Abschnitte enthalten:
    - Einleitung
    - Methodik
    - Ergebnisse
    - Diskussion
    - Fazit
    """

    # API-Aufruf an GPT-4o mit dem korrekten Nachrichtenformat
    response = client.chat.completions.create(
        model="o3-mini",
        messages=[
            {"role": "system", "content": "Du bist ein hilfreicher
Assistent, der wissenschaftliche Berichte strukturiert."},
            {"role": "user", "content": prompt}
        ],
        max_tokens=500  # Maximale Länge der Antwort
    )

    # Extrahiere die Antwort aus der API-Response
    gliederung = response.choices[0].message.content
    print("\nLogische Gliederung:")
    print(gliederung)
    return gliederung

# Schritt 3: Kreative Textgenerierung mit GPT-4-Turbo
def kreative_textgenerierung(gliederung):
    """
    Generiert einen vollständigen wissenschaftlichen Bericht basierend auf
der erstellten Gliederung.
    """
    prompt = f"""
    Schreibe einen wissenschaftlichen Bericht basierend auf der folgenden
Gliederung:

    {gliederung}

    Der Bericht sollte klar, prägnant und wissenschaftlich formuliert sein.
    """

    # API-Aufruf an GPT-4o
    response = client.chat.completions.create(
        model="gpt-4o",
        messages=[
            {"role": "system", "content": "Du bist ein hilfreicher
```

```
Assistent, der wissenschaftliche Berichte schreibt."},
            {"role": "user", "content": prompt}
        ],
        max_tokens=2000  # Maximale Länge des generierten Berichts
    )

    bericht = response.choices[0].message.content
    print("\nWissenschaftlicher Bericht:")
    print(bericht)
    return bericht

# Schritt 4: Vereinfachte multimodale Präsentation mit GPT-4.5
def multimodale_praesentation(daten, client):
    """
    Erstellt Visualisierungen der Daten mit Hilfe von GPT-4.5 und Plotly
Express.
    """
    # Numerische Spalten identifizieren
    numerische_spalten = daten.select_dtypes(include=
['number']).columns.tolist()

    if len(numerische_spalten) < 2:
        raise ValueError("Nicht genügend numerische Spalten für
Visualisierung")

    # Einfache Datenbeschreibung erstellen
    daten_beschreibung = daten[numerische_spalten].describe().to_dict()
    korrelationen = daten[numerische_spalten].corr().to_dict()

    # GPT-4.5 für Visualisierungsvorschläge nutzen
    prompt = f"""
    Analysiere diese Daten und empfiehl eine Visualisierung im JSON-Format:

    Spalten: {numerische_spalten}
    Statistiken: {daten_beschreibung}
    Korrelationen: {korrelationen}

    Rückgabeformat:
    {{
      "diagramm_typ": "scatter/bar/line/histogram/box/heatmap",
      "x_spalte": "spaltenname",
      "y_spalte": "spaltenname oder null",
      "titel": "vorgeschlagener Titel",
      "beschreibung": "kurze Erklärung zur Visualisierung"
    }}
    """

    # API-Aufruf an GPT-4.5
    response = client.chat.completions.create(
        model="gpt-4.5-turbo",  # In der Praxis: "gpt-4o" oder "gpt-4-turbo"
```

```
verwenden
        messages=[
            {"role": "system", "content": "Du bist ein
Datenvisualisierungsexperte. Antworte nur im JSON-Format."},
            {"role": "user", "content": prompt}
        ],
        response_format={"type": "json_object"}
    )

    # JSON-Antwort extrahieren und parsen
    import json
    empfehlung = json.loads(response.choices[0].message.content)

    # Visualisierung basierend auf der Empfehlung erstellen
    diagramm_typ = empfehlung["diagramm_typ"]
    x_spalte = empfehlung["x_spalte"]
    y_spalte = empfehlung.get("y_spalte")
    titel = empfehlung["titel"]

    # Diagramm erstellen
    fig = None

    if diagramm_typ == "scatter":
        fig = px.scatter(daten, x=x_spalte, y=y_spalte, title=titel)
    elif diagramm_typ == "bar":
        fig = px.bar(daten, x=x_spalte, y=y_spalte, title=titel)
    elif diagramm_typ == "line":
        fig = px.line(daten, x=x_spalte, y=y_spalte, title=titel)
    elif diagramm_typ == "histogram":
        fig = px.histogram(daten, x=x_spalte, title=titel)
    elif diagramm_typ == "box":
        fig = px.box(daten, x=x_spalte, y=y_spalte, title=titel)
    elif diagramm_typ == "heatmap":
        if y_spalte:
            pivot_daten = daten.pivot_table(index=x_spalte,
columns=y_spalte, aggfunc='mean')
            fig = px.imshow(pivot_daten, title=titel)
        else:
            fig = px.imshow(daten[numerische_spalten].corr(), title=titel)
    else:
        # Fallback auf einfaches Streudiagramm
        fig = px.scatter(daten, x=numerische_spalten[0],
y=numerische_spalten[1], title="Datenpunkte")

    # Beschreibung als Untertitel hinzufügen
    fig.update_layout(
        annotations=[{
            "text": empfehlung["beschreibung"],
            "xref": "paper", "yref": "paper",
            "x": 0.5, "y": -0.15,
```

```
            "showarrow": False
        }]
    )

    fig.show()
    return fig


# Hauptprogramm: Kaskade ausführen
if __name__ == "__main__":
    # Beispiel-Datensatzpfad (CSV-Datei)
    datei_pfad = "daten.csv"

    # Schritt 1: Datenanalyse durchführen
    daten, zusammenfassung, korrelationen = daten_analyse(datei_pfad)

    # Schritt 2: Logische Strukturierung erstellen
    gliederung = logische_strukturierung(zusammenfassung, korrelationen)

    # Schritt 3: Wissenschaftlichen Bericht generieren
    bericht = kreative_textgenerierung(gliederung)

    # Schritt 4: Multimodale Präsentation erstellen
    fig = multimodale_praesentation(daten)
```

Vorteile einer Modellkaskade

1. **Effizienzsteigerung:** Jedes Modell wird für Aufgaben eingesetzt, die seinen Stärken entsprechen.
2. **Kostenoptimierung:** Ressourcenschonende Modelle wie o3 Mini übernehmen einfache Aufgaben, während leistungsstarke Modelle wie ChatGPT o1 für komplexe Analysen verwendet werden.
3. **Flexibilität:** Die Kombination verschiedener Modelle ermöglicht die Bearbeitung unterschiedlichster Anforderungen.

Fazit

Die Modellkaskade ist besonders nützlich bei Projekten, die unterschiedliche Fähigkeiten erfordern, wie etwa Datenanalyse, logische Strukturierung, kreative Textgenerierung und multimodale Präsentation. Durch die gezielte Nutzung der Modelle können sowohl Kosten als auch Zeit optimiert werden.

6 | Benchmarks & Modellvergleiche

MMLU (Massive Multitask Language Understanding)

Standard-Benchmark für Sprachverständnis über 57 Fachgebiete:

Modell	Score
GPT-4o	88,7%
Gemini 2.0 Ultra	90,0%
Claude 3 Opus	88,2%
Llama 3.1 405B	87,3%
gpt-4o-mini	70,0%

MMLU

Spezialisierte Benchmark-Plattformen

Überblick über wichtige Benchmark-Plattformen im Jahr 2025:

- **LLM Leaderboard**: Bewertet 50+ Modelle nach Kontextfenster, Verarbeitungsgeschwindigkeit und Kosten
- **Hugging Face Open LLM Leaderboard**: Fokus auf Open-Source-Modelle mit Metriken für Textgenerierung und Code-Qualität
- **SEAL Leaderboards (Scale AI)**: Fokus auf Unternehmensanforderungen

Anwendungsbezogene Leistungsvergleiche

Vergleich der Modelle für spezifische Anwendungsfälle:

- **Multimodale Analysen**: Gemini 2.0 Ultra (2M Token Kontext) vs. GPT-4o (128K Token)
- **Echtzeit-Anwendungen**: Claude 3 Haiku: 76,7% MMLU bei 200ms Antwortzeit
- **Kostenoptimierung**: Mistral 7B (32K Token) bei 60,1% MMLU für €0,08/1M Tokens

Praktische Anwendung der Benchmarks

Leitfaden zur Interpretation und Anwendung der Benchmark-Ergebnisse für die eigene Modellauswahl:

- MMLU als Goldstandard für Allgemeinwissen
- SEAL für Unternehmenslösungen
- Hugging Face für Open-Source-Modelle

Diese strukturierte Herangehensweise ermöglicht eine fundierte Entscheidung darüber, welches KI-Modell die gestellten Anforderungen am besten erfüllt und wie es im eigenen Projekt optimal eingesetzt werden kann.

7 A | Aufgabe

Die Aufgabestellungen unten bieten Anregungen, Sie können aber auch gerne eine andere Herausforderung angehen.

Anforderungsanalyse für ein KI-Projekt

Entwickeln Sie eine strukturierte Anforderungsanalyse für ein fiktives oder reales KI-Projekt.

Aufgabenstellung:

1. Wählen Sie einen konkreten Anwendungsfall (z.B. Kundenservice-Chatbot für eine Bank, Content-Generator für Social Media, oder Übersetzungstool für technische Dokumentation).
2. Definieren Sie:
   - Die primären Funktionen, die das KI-Modell erfüllen soll
   - Die spezifischen Anforderungen an das Sprachverständnis
   - Notwendige Fachkenntnisse in relevanten Domänen
   - Anforderungen an die Antwortgeschwindigkeit
   - Budget-Rahmenbedingungen
3. Erstellen Sie eine Prioritätenliste dieser Anforderungen (unbedingt erforderlich, wichtig, wünschenswert).
4. Beschreiben Sie, welche Kompromisse Sie bei konkurrierenden Anforderungen eingehen würden.

Abgabeformat:
Erstellen Sie ein Dokument mit Ihrer Anforderungsanalyse (1-2 Seiten).

Vergleichsanalyse bekannter KI-Modelle

Führen Sie eine vergleichende Analyse von mindestens drei verschiedenen KI-Modellen anhand vorgegebener Bewertungskriterien durch.

Aufgabenstellung:

1. Wählen Sie drei KI-Modelle aus der folgenden Liste aus:
   - GPT-4o
   - Claude 3 Opus
   - Gemini 2.0 Ultra
   - Llama 3.1
   - Mistral 7B
   - Ein anderes aktuelles KI-Modell Ihrer Wahl
2. Recherchieren Sie die Leistungsmerkmale dieser Modelle anhand der folgenden Kriterien:
   - MMLU-Score oder vergleichbare Benchmark-Ergebnisse
   - Kontextfenstergröße
   - Antwortlatenz
   - Kosten (pro Token oder alternativer Maßstab)
   - Verfügbarkeit (API, Open-Source, etc.)
   - Unterstützte Sprachen
   - Multimodale Fähigkeiten (falls vorhanden)
3. Erstellen Sie eine Bewertungstabelle mit den recherchierten Informationen.
4. Verfassen Sie eine begründete Empfehlung, welches dieser Modelle sich für folgende Szenarien am besten eignen würde:
   - Entwicklung eines kostengünstigen Chatbots für ein kleines Unternehmen
   - Erstellung von KI-generierten Inhalten für ein internationales Nachrichtenportal
   - Unterstützung bei der Software-Entwicklung

Abgabeformat:
Vergleichstabelle mit Bewertungen und einer Seite mit Ihren Empfehlungen.

Konzept für die qualitative Evaluation eines Sprachmodells

Entwickeln Sie ein strukturiertes Testverfahren zur qualitativen Bewertung eines Sprachmodells.

Aufgabenstellung:

1. Entwerfen Sie ein Bewertungsschema mit 5-7 qualitativen Kategorien, die für Ihre gewählte Anwendung relevant sind (z.B. Genauigkeit, Kreativität, Nützlichkeit der Antworten, Verständnis komplexer Anweisungen, Kulturelle Sensibilität).
2. Erstellen Sie für jede Kategorie:
   - Eine klare Definition, was in dieser Kategorie bewertet wird
   - Eine Bewertungsskala (z.B. 1-5 oder 1-10)
   - 2-3 konkrete Testfragen oder -aufgaben, die diese Kategorie prüfen
   - Bewertungskriterien: Was wäre eine ausgezeichnete (5/5) vs. eine unzureichende (1/5) Antwort?

3. Beschreiben Sie den Evaluationsprozess:
    - Wie viele Bewerter sollten eingesetzt werden?
    - Wie würden Sie die Bewertungen zusammenfassen?
    - Welche Maßnahmen würden Sie ergreifen, um Bewertungsverzerrungen zu vermeiden?
4. Erläutern Sie, wie Sie die Ergebnisse dieser qualitativen Bewertung mit quantitativen Metriken (wie MMLU) kombinieren würden, um ein Gesamtbild der Modellleistung zu erhalten.

Abgabeformat:
Ein 2-3 seitiges Konzeptpapier mit Ihrem Evaluationsschema, den Testfragen und dem geplanten Prozess.