Projektaufgabe:

Kleine KI-Modelle in der Praxis

Thema:

Implementierung einer smarten To-Do-Liste

mit TinyLlama

Name:

Hideat TEKIE HAILU

Date:

25.08.2025

1. Auswahl der Aufgabenstellung

In diesem Projekt wurde die Aufgabe "Smarte To-Do-Liste" gewählt. Das Ziel ist es, unstrukturierten Text (einen Satz) in eine strukturierte To-Do-Liste umzuwandeln. Als KI-Modell wurde TinyLlama-1.1B-Chat-v1.0 ausgewählt, da es klein, leistungsfähig und ressourcenschonend ist.

2. Recherche

- **Warum ist dieses Modell „klein“ und ressourcensparend?**
  Das Modell TinyLlama ist „klein“, da es nur 1,1 Milliarden Parameter hat. Große Modelle wie GPT-3 haben über 175 Milliarden. Da es weniger Parameter hat, benötigt es weniger Arbeitsspeicher (RAM) und kann auf Standardhardware, wie einem normalen Laptop, ohne leistungsstarke, teure GPUs ausgeführt werden. Das macht es sehr ressourcenschonend.

- **Welche Modellfamilien gibt es aktuell?**
  Es gibt mehrere bekannte Modellfamilien. Die **Llama**-Familie (einschließlich TinyLlama) wird von **Meta** bereitgestellt. Die **Mistral**-Familie stammt von der Firma **Mistral AI**. Die **Phi**-Familie wird von **Microsoft** entwickelt. Viele dieser Modelle werden auf Plattformen wie **Hugging Face** der Öffentlichkeit zugänglich gemacht, das als zentraler Hub für die KI-Community dient.

- **Welche Frameworks/Libraries werden genutzt?**
  Für dieses Projekt ist die primäre Python-Bibliothek **transformers** von Hugging Face. Sie bietet eine einfache Schnittstelle (genannt pipeline), um eine Vielzahl von vortrainierten Modellen herunterzuladen und zu verwenden. Sie stützt sich auf ein Low-Level-Framework wie **PyTorch** oder **TensorFlow** für die eigentlichen Deep-Learning-Berechnungen.

3. Implementierung

➢ **Quellcode (smarte_todo_liste.py):**

```
# Zuerst importieren wir die benötigten Werkzeuge aus der 'transformers'-Bibliothek
from transformers import pipeline
import time

print("--- FINALES PROGRAMM: SMARTE TO-DO-LISTE MIT TINYLLAMA (Final Version) ---")
print("Lade das 'TinyLlama' Modell. Dies kann einen Moment dauern...")

# Wir verwenden das 'TinyLlama' Modell, da es als einziges korrekte Ergebnisse liefert.
generator = pipeline('text-generation', model='TinyLlama/TinyLlama-1.1B-Chat-v1.0')

print("Modell geladen. Generiere jetzt die Antwort...")

# Der unstrukturierte Satz, den wir verarbeiten wollen
messy_sentence = "Ich darf nicht vergessen, morgen den Müll rauszubringen und den Hund zu füttern, und
dann muss ich noch die Hausaufgaben für die Schule machen."

# Der einfache und zuverlässige Prompt, der nachweislich funktioniert
prompt = f"Erstelle aus dem folgenden Satz eine einfache To-Do-Liste mit
Aufzählungszeichen:\n\n'{messy_sentence}'\n\nHier ist die To-Do-Liste:\n-"

# --- START DER ZEITMESSUNG ---
start_time = time.time()

# Wir geben dem Modell genug Platz für die Antwort
results = generator(prompt, max_length=200)

# --- ENDE DER ZEITMESSUNG ---
end_time = time.time()
duration = end_time - start_time


# --- SAUBERE AUSGABE ---
print("\n--- ANTWORT DES KI-MODELLS (TinyLlama) ---")
full_answer = results[0]['generated_text']
try:
    # Schritt 1: Wir extrahieren den Teil nach unserer Anweisung
    todo_list_full = full_answer.split("Hier ist die To-Do-Liste:\n-")[1]

    # Schritt 2: Wir entfernen den zusätzlichen Text, den das Modell generiert hat.
    clean_list = todo_list_full.split("\n\n")[0]

    print("-" + clean_list)

except IndexError:
    # Falls die saubere Ausgabe fehlschlägt, geben wir den Roh-Output aus
    print("Die Ausgabe konnte nicht sauber verarbeitet werden. Roh-Output:")
    print(full_answer)

# Wir geben aus, wie lange die Generierung gedauert hat, um die Langsamkeit zu dokumentieren
print(f"\nGenerierung dauerte: {duration:.2f} Sekunden")```

---
```

Anleitung zum Starten des Programms:

1. Stellen Sie sicher, dass Python 3 auf Ihrem System installiert ist.
2. Erstellen Sie eine virtuelle Umgebung und aktivieren Sie sie:
   python -m venv umgebung
   .\umgebung\Scripts\Activate.ps1
3. Installieren Sie die erforderlichen Bibliotheken:
   pip install transformers torch
4. Führen Sie das Python-Skript aus:
   python smarte_todo_liste.py

Beispielhaftes Ergebnis

ANTWORT DES KI-MODELLS (TinyLlama)

- 1: Morgen den Müll rauszubringen

- 2: Hund zu füttern

- 3: Hausaufgaben für Schule

4. Reflexion

Laufzeit: Die Generierung dauerte 611.78 Sekunden.

➢ **Welche Einschränkungen haben kleine Modelle?**

In diesem Projekt wurde der Kompromiss zwischen Geschwindigkeit und Qualität deutlich.

- **Qualität & Genauigkeit:** Das TinyLlama-Modell war intelligent genug, um die deutsche Sprache zu verstehen und die Aufgabe korrekt zu lösen. Ein Test mit dem kleineren, schnelleren Modell distilgpt2 schlug jedoch fehl. Es konnte die Anweisung nicht verstehen und produzierte unsinnigen Text. Dies zeigt, dass die Genauigkeit kleinerer Modelle für spezifische Aufgaben, insbesondere in anderen Sprachen als Englisch, stark eingeschränkt sein kann.
- **Geschwindigkeit:** Die größte Einschränkung des funktionierenden TinyLlama-Modells war die Geschwindigkeit. Auf einer CPU dauerte die Generierung der Antwort über 10 Minuten (ca. 611 Sekunden), was für eine Echtzeitanwendung zu langsam ist.

➢ **Was sind geeignete Anwendungsbereiche?**

- **Geeignet für: Spezifische, einfache Aufgaben wie Textklassifizierung (z.B. Sortieren von Support-Tickets), Stimmungsanalyse (positive/negative Bewertungen), einfache Chatbots und die Erstellung von strukturiertem Text aus unstrukturiertem Text (wie unsere To-Do-Liste!). Sie sind auch ideal für Anwendungen, die offline auf Geräten wie einem Raspberry Pi laufen müssen.**
- **Ungeeignet für: Komplexe Aufgaben, die tiefes logisches Denken erfordern, wie das Verfassen einer wissenschaftlichen Arbeit, fortgeschrittene Programmierung oder die Bereitstellung von medizinischem Fachrat.**