



# Tic-Tac-Toe mit Llama3.2

**Tumar Chegebaeva**

Multimodale Mensch Computer Interaktion

July 14, 2025

# Übersicht



## 1. Implementierung

## 2. Ergebnisse

## 3. Zusammenfassung



# Implementierung

---

# Einfaches Tic-Tac-Toe

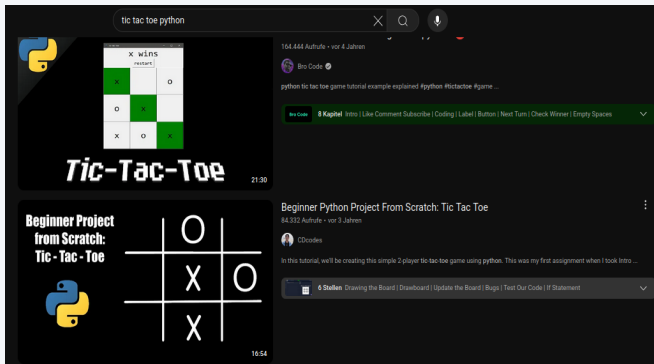


Figure: Grundgerüst nach Video-Tutorials



## Woche 2: Übungen

- Sentimentanalyse: Stellen Sie sich vor, dass Sie eine Einschätzung von Studierendenkommentaren zu einer Vorlesung haben möchten. Die Kommentare (Reviews) sollen in 3 Kategorien eingeteilt werden: positiv, neutral, negativ
- Entwickeln Sie einen Prompt, der das LLM dazu bringt, einen übergebenen Review zu analysieren und in eine entsprechende Klasse einzuordnen.
  - Geben Sie im Prompt Beispiele für die jeweiligen Klassen
  - Strukturieren Sie den Prompt (durch Überschriften)
  - Geben Sie dem System Hinweise zur Ordnung der Kategorien
  - Als Ergebnis der Analyse soll lediglich die Kategorie ausgegeben werden
  - Speichern Sie den Prompt in einer Datei „meinePrompts.txt“, der Anfang Ihrer persönlichen Sammlung an guten Prompts

Figure: Erste Überlegung zum Prompt-Design

# Anbindung des Sprachmodells



```
# 1) Konstruktion des strukturierten Prompts
prompt = f"""
Analysiere den folgenden Studierendenkommentar zu einer Vorlesung und ordne ihn in eine der drei Kategorien ein: p
Hinweise zur Kategorisierung:
- positiv: Lob/Begeisterung (z.B. „Toll erklärt!“)
- neutral: Sachlich oder gemischt (z.B. „Inhalt okay, aber zu schnell“)
- negativ: Kritik/Ablehnung (z.B. „Unorganisiert und unverständlich“)
REVIEW:{text}
"""

##### 2) Konfiguration des Systems #####
# System prompt: Hier beschreibt man genau, welche Rolle das System einnehmen soll. Es konfiguriert das Verhalten.
system_message = {
    "role": "system",
    "content": "Du bist ein hilfreicher KI-Assistent, der eine Sentimentanalyse durchführen soll."
}

# User prompt: Das ist die eigentliche Anfrage, die der Anwender an das System stellt.
user_message = {
    "role": "user",
    "content": f"{prompt}"
}

messages = [system_message, user_message]

##### Ende Konfiguration #####

## 3) Die eigentliche Anfrage an das LLM über ollama stellen
result = ollama.chat(
    model='llama3.2',
    messages=messages
)

## 4) Ausgabe des Antwort
print("Ergebnis:", result['message']['content'])
```

Figure: Beispielhafte Prompt-Konfiguration

```

1 import os
2 import ollama
3
4 board = [' ']*9
5
6 def display_board():
7     print(f"""
8         -----
9         | {board[0]} | {board[1]} | {board[2]} |
10        -----
11        | {board[3]} | {board[4]} | {board[5]} |
12        -----
13        | {board[6]} | {board[7]} | {board[8]} |
14        -----
15    """)
16
17 def check_winner():
18     wins = [
19         [0, 1, 2], [3, 4, 5], [6, 7, 8],
20         [0, 3, 6], [1, 4, 7], [2, 5, 8],
21         [0, 4, 8], [2, 4, 6]
22     ]
23     for a,b,c in wins:
24         if board[a] == board[b] == board[c] and board[a] != ' ':
25             return True
26     return False
27
28 # Board für den KI-Prompt
29 def board_to_string():
30     return "\n" + "\n".join([
31         f"{board[0]} | {board[1]} | {board[2]}",
32         f"{board[3]} | {board[4]} | {board[5]}",
33         f"{board[6]} | {board[7]} | {board[8]}"
34     ])
35
36 # Anfrage an Llama3.2 schicken für den KI-Zug als 0
37 def get_ai_move():
38
39     prompt = f"""
40         Hier ist das Spielbrett (X = Mensch, 0 = du): {board_to_string()}
41         Deine Aufgabe:
42         Gib bitte nur eine Zahl von 1 bis 9 zurück. Wähle eine Zahl, die noch nicht belegt
43         ist.
44         """
45
46     system_message = {
47         "role": "system",
48         "content": "Du bist ein Tic Tac Toe-Spieler als 0. Wähle die nächste freie Position von 1 bis 9, um zu gewinnen oder zu blockieren."
49     }
50
51     user_message = {
52         "role": "user",
53         "content": prompt
54     }
55
56     messages = [system_message, user_message]
57     response = ollama.chat(model="llama3.2", messages=messages)
58     content = response['message']['content'].strip()
59
60     # Zahl zu extrahieren
61     try:
62         move = int(''.join(filter(str.isdigit, content)))
63         return move
64     except:
65         return None
66
67 turn = 0
68 game_running = True
69 has_winner = False

```

```

70 while game_running:
71     os.system('cls' if os.name == 'nt' else 'clear')
72     display_board()
73
74     current_player = 'X' if turn % 2 == 0 else 'O'
75
76     if current_player == 'X':
77         try:
78             move = int(input("Dein Zug 1-9: "))
79             if move < 1 or move > 9 or board[move - 1] != ' ':
80                 print("Ungültiger Zug. Bitte erneut.")
81                 input()
82                 continue
83         except ValueError:
84             print("Bitte gib eine Zahl ein.")
85             input()
86             continue
87     else:
88         print("KI denkt...")
89         move = get_ai_move()
90         while move is None or move < 1 or move > 9 or board[move - 1] != ' ':
91             print(f"Ungültiger KI-Zug {move}. Versuche erneut...")
92             move = get_ai_move()
93         print(f"KI wählt Feld: {move}")
94
95     board[move - 1] = current_player
96     turn += 1
97
98     if check_winner():
99         has_winner = True
100         game_running = False
101     elif turn == 9: game_running = False
102
103 # Finale Anzeige
104 os.system('cls' if os.name == 'nt' else 'clear')
105 display_board()
106
107 if has_winner:
108     winner = 'O' if turn % 2 == 0 else 'X'
109     print(f"Spieler {winner} gewinnt!")
110 else:
111     print("Unentschieden.")
112
113 print("Danke fürs Spielen!")

```





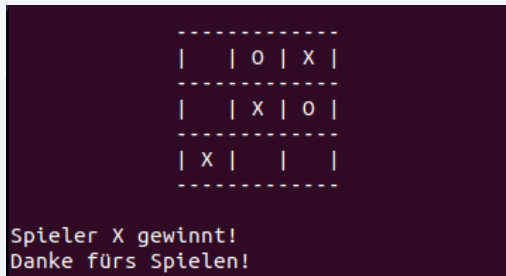
# Ergebnisse

---

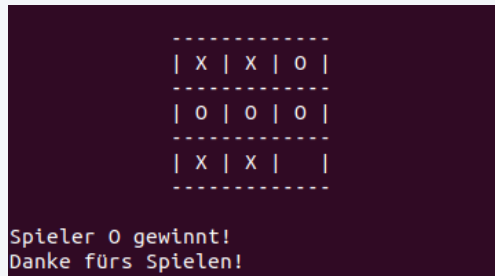


# Ergebnisse der Spielrunden

Spieler X gewinnt



Spieler O gewinnt





# Verzögerungen bei der KI-Zugauswahl

In seltenen Fällen  
benötigte Llama3.2  
bis 30 Sekunden für  
eine gültige Antwort.  
Typischerweise lag  
die Reaktionszeit bei  
2-4 Sekunden pro Zug.

```

  -----
  |   |   | x |
  |   | o | o |
  | x |   | x |
  -----
KI denkt...
Ungültiger KI-Zug 33. Versuche erneut...
Ungültiger KI-Zug 33. Versuche erneut...
Ungültiger KI-Zug 3. Versuche erneut...
Ungültiger KI-Zug 3. Versuche erneut...
Ungültiger KI-Zug 33. Versuche erneut...
Ungültiger KI-Zug 3. Versuche erneut...
Ungültiger KI-Zug 3. Versuche erneut...
Ungültiger KI-Zug 3. Versuche erneut...
Ungültiger KI-Zug 3. Versuche erneut...
Ungültiger KI-Zug 33. Versuche erneut...
Ungültiger KI-Zug 3. Versuche erneut...
Ungültiger KI-Zug 33. Versuche erneut...
Ungültiger KI-Zug 3. Versuche erneut...
Ungültiger KI-Zug 3. Versuche erneut...
Ungültiger KI-Zug 3. Versuche erneut...
Ungültiger KI-Zug 33. Versuche erneut...
Ungültiger KI-Zug 3. Versuche erneut...
Ungültiger KI-Zug 33. Versuche erneut...
Ungültiger KI-Zug 6. Versuche erneut...
Ungültiger KI-Zug 3. Versuche erneut...
Ungültiger KI-Zug 6. Versuche erneut...
Ungültiger KI-Zug 6. Versuche erneut...
Ungültiger KI-Zug 6. Versuche erneut...
Ungültiger KI-Zug 33. Versuche erneut...
Ungültiger KI-Zug 33. Versuche erneut...
Ungültiger KI-Zug 6. Versuche erneut...
Ungültiger KI-Zug 6. Versuche erneut...
Ungültiger KI-Zug 3. Versuche erneut...
Ungültiger KI-Zug 33. Versuche erneut...
Ungültiger KI-Zug 3. Versuche erneut...
Ungültiger KI-Zug 3. Versuche erneut...
Ungültiger KI-Zug 3. Versuche erneut...
Ungültiger KI-Zug 3. Versuche erneut...
```



# Zusammenfassung

---



# Schlussfolgerung

- Llama3.2 ist primär ein Sprachmodell ohne Spielstrategie.
- API-Aufrufe führten zu verzögerten Antwortzeiten.
- Es gab keine Implementierung für einen vorzeitigen Spielabbruch.

# Referenzen



Jan-Torsten Milde (2025)

schueler\_tag\_KI

[https://github.com/drmilde/schueler\\_tag\\_KI](https://github.com/drmilde/schueler_tag_KI)



OpenAI (2025)

ChatGPT: A Large Language Model by OpenAI

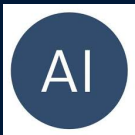
<https://openai.com/chatgpt>



YouTube

Verschiedene Tutorials zu Python und Tic-Tac-Toe

<https://www.youtube.com>



# Fragen ? Stellen : Danke

**Tumar Chegebaeva**

Multimodale Mensch Computer Interaktion

July 14, 2025