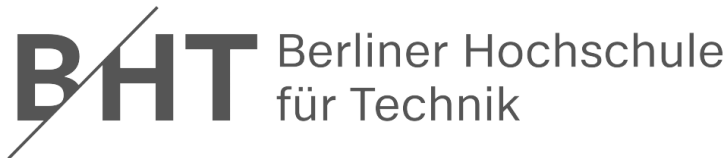


Evaluierung und Optimierung von Large Language Models für die Entwicklung von Webanwendungen

Ein Ansatz zur Verbesserung des Entwicklungsprozesses bei Softwareprojekten



Masterthesis
für den Master of Science Studiengang Medieninformatik

eingereicht von: Wilfried Pahl
Matrikelnummer: 901932
Studiengang: Online Medieninformatik
Berliner Hochschule für Technik

betreut durch Prof. Dr. S. Edlich
Berliner Hochschule für Technik

Temmen-Ringenwalde, der 16. Oktober 2024

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit mit dem Titel „Evaluierung und Optimierung von Large Language Models für die Entwicklung von Webanwendungen (*Ein Ansatz zur Verbesserung des Entwicklungsprozesses bei Softwareprojekten*)“ selbstständig und ohne unerlaubte Hilfe verfasst habe. Alle benutzten Quellen und Hilfsmittel sind vollständig angegeben und wurden entsprechend den wissenschaftlichen Standards zitiert.

Ich versichere, dass alle Passagen, die nicht von mir stammen, als Zitate gekennzeichnet wurden und dass alle Informationen, die ich aus fremden Quellen übernommen habe, eindeutig als solche kenntlich gemacht wurden. Insbesondere wurden alle Texte und Textpassagen anderer Autoren sowie die Ergebnisse von Sprachmodellen wie OpenAI's GPT-3 entsprechend den wissenschaftlichen Standards zitiert und referenziert.

Ich versichere weiterhin, dass ich keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe und dass ich keine Teile dieser Arbeit in anderer Form für Prüfungszwecke vorgelegt habe.

Mir ist bewusst, dass eine falsche eidesstattliche Erklärung strafrechtliche Konsequenzen haben kann.

Temmen-Ringenwalde, den 16. Oktober 2024

Unterschrift

ABSTRACT

Abstract in Englisch.

Entwurf

ZUSAMMENFASSUNG

Zusammenfassung in Deutsch.

Entwurf

INHALTSVERZEICHNIS

Abstract	i
Abbildungsverzeichnis	v
Tabellenverzeichnis	vi
Listings	viii
Abkürzungsverzeichnis	x
1 Einleitung	1
1.1 Hintergrund und Kontext	1
1.2 Problemstellung	2
1.3 Zielsetzung und Forschungsfragen	2
1.4 Aufbau der Arbeit	2
1.5 Abgrenzung	3
2 Grundlagen	5
2.1 Künstliche Intelligenz	5
2.1.1 Historisches	6
2.1.2 Maschinelles Lernen	6
2.1.3 Lernparadigmen des ML	8
2.1.4 Theoretische Grundlagen	9
2.1.5 Neuronale Netze	10
2.1.6 Deep Learning	10
2.1.7 Natural Language Processing	10
2.2 Large Language Model	10
2.2.1 Grundlagen	11
2.2.2 Historie der LLM	11
2.3 Orchestrierung von LLMs	11

2.4	Multi-Agenten-Systeme	11
2.5	Prompt Engineering	11
2.5.1	Prompt-Techniken	12
2.5.2	Grenzen beim Prompt-Engineering für LLMs	28
2.6	Grundlagen bei der Entwicklung von Webanwendungen	28
3	Stand der Forschung	29
3.1	Methoden und Ansätze	29
3.2	Forschungslücken und zukünftige Forschung	29
3.2.1	Identifikation von Forschungslücken	29
3.2.2	Zukünftige Forschungsrichtungen	29
4	Methodik	31
4.1	Auswahl der LLM	31
4.2	Prompt-Engineering	31
5	Implementierung	33
5.1	Modelle lokal aufsetzen	33
5.1.1	Install Ollama	33
5.1.2	Open WebUI	34
5.1.3	Python Client	34
6	Evaluation	35
6.1	Einfache HTML Seite	35
6.1.1	ChatGPT 3.5	35
7	Anwendungsszenarien	37
8	Diskussion und Ausblick	39
9	Fazit	41
	Literatur	43
	Glossar	44
	Anhang	45

ABBILDUNGSVERZEICHNIS

2.1	Einordnung LLMs	5
2.2	KI Winterzyklen	7
2.3	Lernparadigmen des maschinellen Lernens	8
2.4	Baumstruktur der „Tree of Thoughts“ Technik	26

Entwurf

TABELLENVERZEICHNIS

Entwurf

Entwurf

LISTINGS

2.1	Zero-Shot Prompt als Python-String	12
2.2	Antwort des Zero-Shot-Prompts	12
2.3	Few-Shot Prompt als Python-String	13
2.4	Antwort des Few-Shot-Prompts	14
2.5	CoT Prompt als Python-String	15
2.6	Antwort des CoT-Prompts	15
2.7	Meta Prompt als Python-String	17
2.8	Antwort des Meta-Prompts	18
2.9	Chain Prompt Nr. 1 als Python-String	20
2.10	Antwort des Chain-1-Prompts	20
2.11	Chain Prompt Nr. 2 als Python-String	21
2.12	Antwort des Chain-2-Prompts	22
2.13	Chain Prompt Nr. 3 als Python-String	23
2.14	Antwort des Chain-3-Prompts	23
2.15	Ausgabe für DOMPDF Bibliothek	25
2.16	Ausgabe für MPDF Bibliothek	26
2.17	Ausgabe für TCPDF Bibliothek	27
5.1	Ollama Hostanpassng für Netzwerkbetrieb	33

Entwurf

ABKÜRZUNGSVERZEICHNIS

CoT Chain of Thoughts.

k-NN k-Nearest Neighbors.

KI Künstliche Intelligenz.

LLM Large Language Model.

ML Maschine Learning.

ToT Tree of Thoughts.

1.1 Hintergrund und Kontext

Durch die zunehmende Globalisierung und Digitalisierung wird die Gesellschaft der Gegenwart und Zukunft geprägt. Der Ausbau von Hochgeschwindigkeitsnetze und die globale Corona-Pandemie haben diese Entwicklung noch einmal beschleunigt. Immer mehr Unternehmen erkennen die Potenziale der Digitalisierung und stellen ihre Prozesse um. Ganze Wertschöpfungsketten werden auf cloudbasierte Umgebungen umgestellt. Angefangen bei der Kommunikation, über Beschaffung und Produktion bis zum Verkauf der Waren und Dienstleistungen. In allen Stufen der Prozesse kommen webbasierte Anwendungen zum Einsatz, um die Kommunikation der Anwender mit den Systemen zu ermöglichen oder Schnittstellen für die Datenübertragung zwischen den Systemen zu gewährleisten. Durch wachsende Anzahl von Web-Anwendungen wächst auch der Druck an die Entwicklungsfirmen, ihre Anwendungen den schnell wechselnden Kundenanforderungen anzupassen (Beweis fehlt).

Durch diesen Prozess getrieben, müssen Entwicklungsfirmen in immer kürzeren Release-Zyklen Softwarekomponenten hinzufügen und vorhandene erweitern. Gleichzeitig wachsen aber auch die Anforderungen an Stabilität und Sicherheit der cloudbasierten Anwendungen, sowie der Bedarf an kostengünstigeren IT-Abläufen (Beweis fehlt). Ein weiteres Problem ist der wachsende Fachkräftemangel in der Wirtschaft und die damit verbundenen steigenden Gehälter der Entwickler (Beweis fehlt).

Die Verwendung künstlicher Intelligenz bei der Programmierung gewinnt immer mehr an Bedeutung. Eine Technologie die im besonderen Maße an dieser Entwicklung beteiligt ist, sind die Large Language Models. Insbesondere mit der Veröffentlichung vom ChatGPT wurde hier ein regelrechter Hype um die LLMs ausgelöst. Diese Modelle erlauben eine Softwareentwicklung mit natürlicher

Sprache. Tiefe Kenntnisse der verwendeten Programmiersprache sind nicht mehr in dem Maße erforderlich, wie ohne LLMs.

1.2 Problemstellung

So groß der Hype um Künstliche Intelligenz auch sein mag, zurzeit kann KI noch nicht alles. Dies sollte auch bei der Verwendung von KI generiertem Inhalten und Code beachtet werden.

KI denkt nicht, KI trifft keine Entscheidungen. Eine KI antwortet auf eine Eingabe nicht mit der besten Antwort, sondern mit der Wahrscheinlichsten.

Test Der Mensch muss die Ergebnisse prüfen, ehe generierte Programmcodestücke in vorhandene Programme eingefügt und in Produktionsumgebungen implementiert werden.

Viele Entwickler setzen auf ChatGPT zur Generierung von Code, wie eine Umfrage von stackoverflow vom Mai 2024 zeigt [1]. Gleichzeitig wachsen auch die technischen Schulden bei Softwareprojekten, da diese Modelle nicht für die Entwicklung von Software optimiert sind (Beweis fehlt).

1.3 Zielsetzung und Forschungsfragen

Diese Arbeit soll eine Auswahl von Modellen evaluieren und dessen Brauchbarkeit für die Softwareentwicklung aufzeigen. Die Modellauswahl wird von der Seite EvalPlus Leaderboard abgeleitet. Hier werden Modelle gewählt, welche erstere Plätze belegen, aber zum Vergleich auch Modelle aus dem Mittelfeld.

Des Weiteren soll gezeigt werden, ob die automatisierte Verwendung beider Techniken eine Effizienz und Effektivität des Entwicklungsprozesses gesteigert werden kann.

1.4 Aufbau der Arbeit

Ein paar Worte zum Aufbau dieser Arbeit. Im Kapitel 3 wird der aktuelle Stand der Forschung vorgestellt und Erkenntnisse anderer Arbeiten diskutiert. Die in dieser Arbeit verwendeten Methoden, werden im Kapitel 4 behandelt. Die Implementierung der Test LLMs wird in Kapitel 5 besprochen und in Kapitel 6 die Ergebnisse evaluiert. Bevor in Kapitel 9 auf mögliche Folgearbeiten eingegangen wird, gibt es in Kapitel 7 Anwendungsszenarien, die zu den Ergebnissen dieser Arbeit geführt haben.

1.5 Abgrenzung

Ausschluss anderer Anwendungsbereiche.

Rechtliche und ethische Überlegungen werden nur am Rande berücksichtigt.

Entwurf

Entwurf

Die hier besprochenen Grundlagen gehen nicht in eine Tiefe, um alle evtl. Fragen zu klären. Jedes einzelne Gebiet könnte eine Arbeit füllen. Stattdessen sollen lediglich einen kleinen Einblick geben.

2.1 Künstliche Intelligenz

Die Grafik 2.1 soll die Einordnung von LLMs im Bereich der künstlichen Intelligenz zeigen.

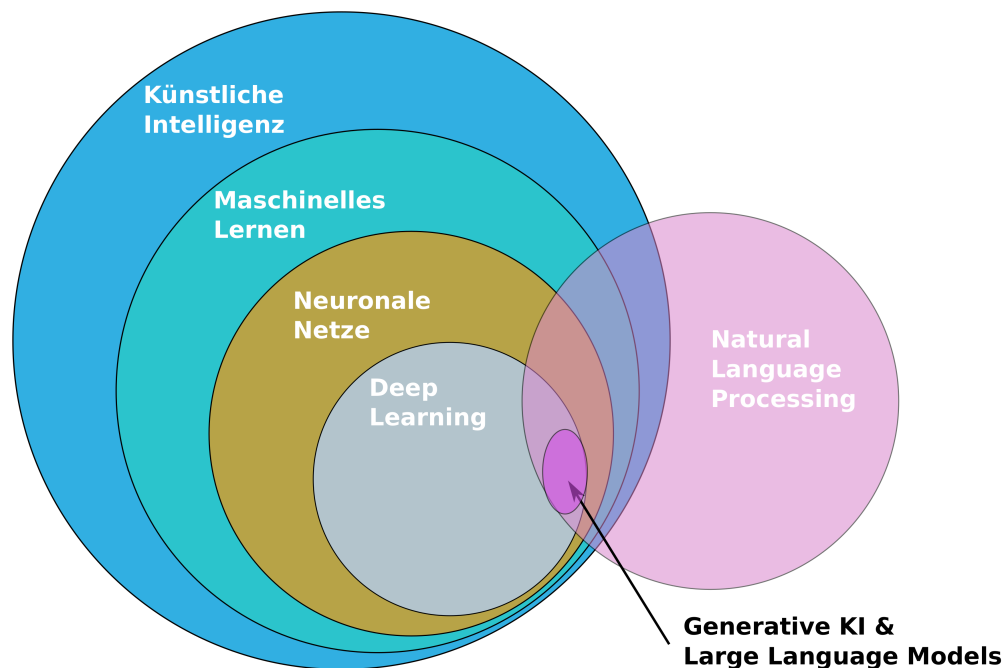


Abbildung 2.1: Einordnung LLMs

In den folgenden Kapiteln werden die wichtigsten Technologien und Begriffe erläutert.

Eine explizite Definition für künstliche Intelligenz ist zurzeit noch nicht erfolgt. Geschuldet ist diese Tatsache, da der Begriff Intelligenz nicht eindeutig definiert ist. Somit finden sich viele Versuche eine Definition für künstliche Intelligenz herzuleiten. In dieser Arbeit wird die Definition aus [2, 6 ff.] verwendet.

Systeme der künstlichen Intelligenz (KI-Systeme) sind vom Menschen entwickelte Softwaresysteme (und gegebenenfalls auch Hardwaresysteme)³, die in Bezug auf ein komplexes Ziel auf physischer oder digitaler Ebene handeln, indem sie ihre Umgebung durch Datenerfassung wahrnehmen, die gesammelten strukturierten oder unstrukturierten Daten interpretieren, Schlussfolgerungen daraus ziehen oder die aus diesen Daten abgeleiteten Informationen verarbeiten, und über das bestmögliche Handeln zur Erreichung des vorgegebenen Ziels entscheiden. KI-Systeme können entweder symbolische Regeln verwenden oder ein numerisches Modell erlernen, und sind auch in der Lage, die Auswirkungen ihrer früheren Handlungen auf die Umgebung zu analysieren und ihr Verhalten entsprechend anzupassen. (Wirtschaftliche Bedeutung ..., 26 ff.).

2.1.1 Historisches

An dieser Stelle eine kleine historische Exkursion in der Entwicklung der künstlichen Intelligenz.

1966: Rückschläge bei der maschinellen Übersetzung

1969: Auswirkungen der Perzeptron-Kritik

1971-75: Die Finanzierungsherausforderungen der DARPA

1973: Der Fallout des Lighthill-Berichts

1973-74: Kürzung der DARPA-Mittel

1987: Zusammenbruch der LISP-Maschine

1988: Kürzungen bei der strategischen Datenverarbeitung

1990er Jahre: Niedergang von Expertensystemen

1990er Jahre: Ende des Projekts der fünften Generation

KI-Frühling des frühen 21. Jahrhunderts

2.1.2 Maschinelles Lernen

Das Gebiet um maschinelles Lernen (eng. Machine Learning) ist ein Teilgebiet der Künstlichen Intelligenz. Für das maschinelle Lernen wird in dieser Arbeit die allgemein gültige Definition nach Tom M. Mitchell

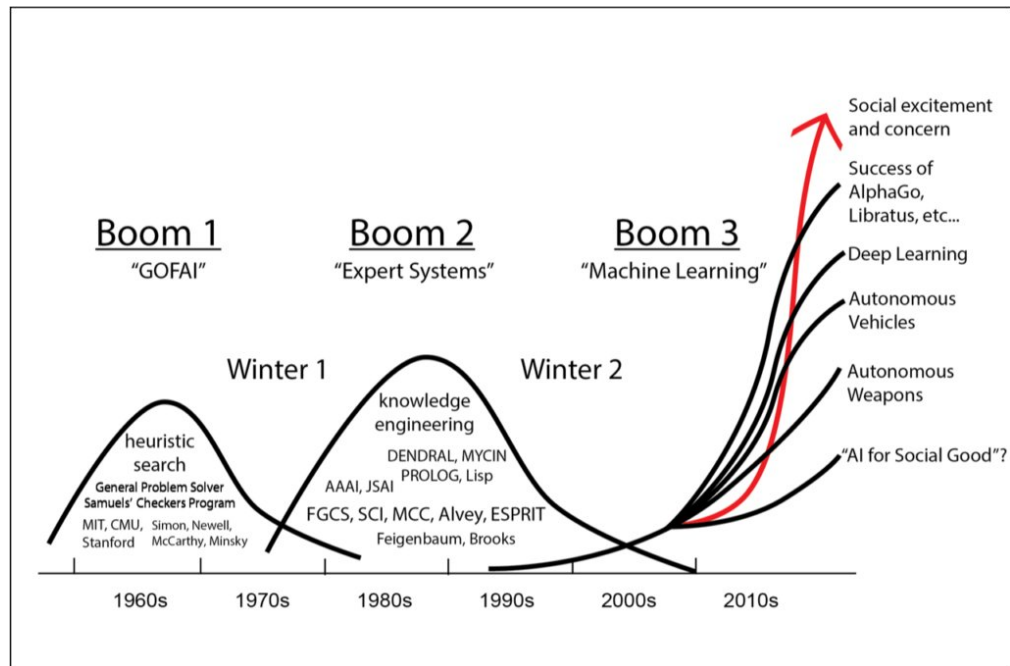


Abbildung 2.2: KI Winterzyklen

verwendet.

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .

Ein Computerprogramm lernt aus Erfahrung E in Bezug auf eine Klasse von Aufgaben T und ein Leistungsmaß P , wenn sich seine Leistung bei Aufgaben in T , gemessen an P , mit der Erfahrung E verbessert.

Mitchell, Tom M

MACHINE LEARNING

Bei dieser Definition ist,

E die **Erfahrung**, die Daten aus denen das System lernt,

T beschreibt die **Aufgabe**, die das System erledigen soll und

P ist die **Leistungskennzahl** an dem der Erfolg des Systems die Aufgabe zu lösen gemessen wird.

Maschinelles Lernen (ML) und Künstliche Intelligenz (KI) sind nicht wirklich in der Lage selbstständig zu lernen oder denken, sie imitieren dies lediglich nach. Das maschinelle Lernen ist aber wohl in der Lage komplexe Muster und Funktionen in großen Datenmengen zu erkennen. Durch die Unfähigkeit zu lernen kann KI keine neuen Inhalte schaffen.

Es ist auch egal wie gut die Modelle trainiert werden, eine 100% Fehlerfreiheit gibt es nicht. Für die Praxis bedeutet dies, die Ausgaben von Modellen müssen durch Menschen immer wieder evaluiert werden, um dessen Richtigkeit sicherzustellen. Zudem sollten keine Modelle verwendet werden, wenn die Lösung eines Problems durch einen konkreten Algorithmus erfolgen kann. Hier können die Ergebnisse nachvollzogen werden und sind für den Menschen besser zu verstehen.

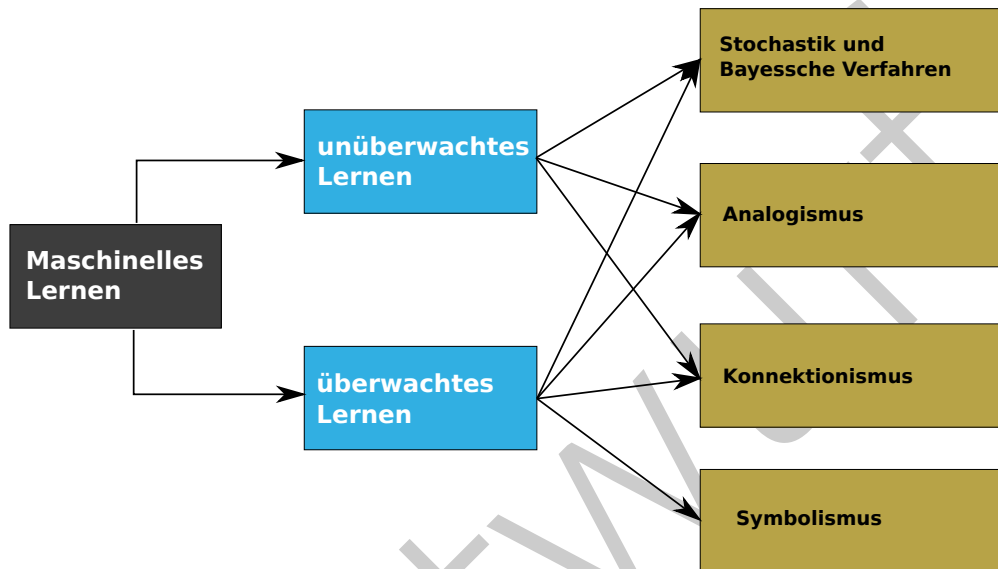


Abbildung 2.3: Lernparadigmen des maschinellen Lernens

2.1.3 Lernparadigmen des ML

Das maschinelle Lernen wird in zwei wichtige Formen der Lernparadigmen unterteilt. Dem überwachten und dem unüberwachten Lernen. Abbildung 2.3 zeigt die wichtigsten Lernparadigmen des maschinellen Lernens.

Überwachtes Lernen

Bei überwachtem Lernen sind für die Eingaben der Trainingsdaten dazugehörige Ausgaben (Labels) definiert. Das Ziel ist es eine Funktion zu trainieren um künftige Eingaben korrekt klassifizieren oder vorhersagen zu können.

Unüberwachtes Lernen

Bei unüberwachten Lernen sind die gelabelten Ausgaben nicht vorhanden. Hierbei wird beispielsweise durch Clustering oder Dimensionsreduktion versucht Muster und Strukturen zu erkennen.

2.1.4 Theoretische Grundlagen

In den folgenden Kapiteln werden vier wichtige Konzepte zum maschinellen Lernen besprochen.

Stochastik und Bayessche Verfahren

Als Teilgebiet der Mathematik befasst sich die Stochastik mit Wahrscheinlichkeiten und zufälligen Prozessen. Auf dem Gebiet des maschinellen Lernens werden mithilfe der Stochastik Prognosen erstellt.

Bei den Bayesschen Verfahren handelt es sich um stochastische Methoden, die auf dem Bayes-Theorem basieren. Hierbei werden neue Daten berücksichtigt, um die Wahrscheinlichkeit zu aktualisieren.

Beide Methoden finden im überwachten und unüberwachten Lernen Anwendung.

Analogismus

Dieser Lernansatz sucht nach Ähnlichkeiten in den Daten. Er basiert auf der Annahme, dass ähnliche Daten ähnliche Vorhersagen oder Klassifizierungen besitzen. Dieses Modell lernt, indem neue Daten mit bekannten verglichen und nach ähnlichen Strukturen und Mustern gesucht. Ein bekanntes Verfahren für diesen Lernansatz ist der k-Nearest Neighbors (k-NN).

Der Analogismus wird im überwachten Lernen als auch im unüberwachten Lernen angewandt, um Muster und Strukturen zuerkennen.

Konnektionismus

Der Lernansatz des Konnektionismus beruht auf kleinen Einheiten, die miteinander verbunden sind. Diese werden als Neuronen bezeichnet, die den Nervenzellen von Organismen nachempfunden sind. Die künstlichen neuronalen Netze sind die bekanntesten Vertreter, auf denen auch Deep Learning Modelle basieren.

Auch dieser Lernansatz wird im unüberwachten und überwachten Lernen angewandt.

Symbolismus

Anders als beim Konnektionismus arbeiten die Einheiten beim Symbolismus mit explizite formale Regeln und Symbole, um das Wissen darzustellen. Der Symbolismus ist weniger flexible im Umgang mit unvollständigen Datensätzen und Unsicherheiten. Daher hat dieser Lernansatz weniger Relevanz als der Konnektionismus.

Dieser Lernansatz findet im überwachten Lernen Anwendung, als Beispiel sind hier Entscheidungsbäume zu nennen.

2.1.5 Neuronale Netze

KNN

Neuronen im neuronalen Netz

Neuronen

Arten der neuronalen Netzen

KNN Arten

Lernprozess im neuronalen Netz / Training

Training

2.1.6 Deep Learning

DL

2.1.7 Natural Language Processing

NPL

2.2 Large Language Model

Large Language Model

2.2.1 Grundlagen

Grundlagen

Tokenisierung

Token

Embedding

Embedding

Vorhersage

Transformer

Dekodierung

Dekodierung

2.2.2 Historie der LLM

Historie

2.3 Orchestrierung von LLMs

Orchestrierung

2.4 Multi-Agenten-Systeme

Multi-Agent-System

2.5 Prompt Engineering

Prompt Engineering optimiert die Antworten großer Sprachmodelle, ohne Parameter, wie Bias und Gewichte des Modells ändern zu müssen. Dieser Bereich hat in den letzten Jahren enorm an Bedeutung gewonnen und sich zu einer eigenen Disziplin im Bereich der Künstlichen Intelligenz entwickelt.

Ein Prompt oder Anweisung muss entweder als Anweisung oder als Frage gestellt werden. Dies kann, wie in [3] beschrieben, in Form von einer einfachen Anweisung bis hin zu detaillierten Beschreibungen oder spezifischen Aufgaben erfolgen.

[Hier Beispiel von ChatGPT oder Gemini einfügen, kann als Bild]

2.5.1 Prompt-Techniken

Siehe Prompting Techniques Hinweise für die Optimierung von Prompts. Die folgenden Techniken dienen dazu die Abfragen zu optimieren und somit eine bessere Antwort von den Sprachmodellen zu erhalten.

Zero-shot Prompting

Bei dieser Technik handelt es sich um das Senden einer einfachen, klaren und präzisen Anweisungen, ohne Angabe von Beispielen und sonstigen zusätzlichen Informationen an die Modelle. Hierbei handelt es sich meistens um eine Domain-spezifische Anweisung. Es findet kein explizites Training vorher statt. Die Anweisung sollte ein klar definiertes Ziel haben

```
1 prompt_string: str = """
Generiere eine JavaScript-Funktion, die ein DIV Tag sucht und einen Text
  einfügt.
  """
```

Listing 2.1: Zero-Shot Prompt als Python-String

Im Folgenden die Antwort vom Modell. Hierbei wurde das Modell *deepseek-coder-v2* verwendet und über die API abgefragt.

```
2 Natürlich! Hier ist ein Beispiel für eine JavaScript-Funktion, die ein '
  DIV'-Element sucht und Text hinzufügt:
  '''javascript
function addTextToDiv(selector, text) {
  // Das Element mit dem angegebenen Selektor suchen
  const divElement = document.querySelector(selector);
  7
  // Wenn das Element gefunden wurde, fügen Sie den Text hinzu
  if (divElement) {
    divElement.innerText = text;
  } else {
```

```

12     console.log("Das DIV-Element mit dem angegebenen Selektor wurde
        nicht gefunden.");
    }
}

// Beispielaufruf der Funktion
17 addTextToDiv('div', 'Hallo, dies ist ein eingefügter Text.');
```

'''

In diesem Code:

- 'selector' ist der CSS-Selektor für das 'DIV'-Element.
- 22 - 'text' ist der Text, den Sie dem 'DIV' hinzufügen möchten.
- Die Funktion verwendet 'document.querySelector(selector)' um das Element zu finden.
- Wenn das Element gefunden wird, fügt die Funktion 'innerText' des Elements den angegebenen Text hinzu.

Listing 2.2: Antwort des Zero-Shot-Prompts

Um bessere Lösungen von den Modellen zu bekommen, kann es sinnvoll sein, weitere Angaben, zum Beispiel zur Arbeitsweise oder die Definition, der zu verwendenden Bibliotheken hinzuzufügen. In [4] wird eine Methode vorgeschlagen, zur Verbesserung der Zero-shot Anweisungen.

Few-shot Prompting

Bei komplexen Aufgaben liefert die Verwendung von Zero-shot Anweisungen oft unzureichende Ergebnisse. Hierfür finden Few-shot Prompts Verwendung. Bei dieser Technik werden ein oder mehrere Beispiele einer Antwort der Anweisung beigefügt, die als eine Art Antwortvorlage für das Modell dienen. Das Listings ?? zeigt beispielhaft den Prompt als Python-String, welcher an das Modell übertragen wurde.

```

1 prompt_string: str = """
Hier sind zwei Beispiele eine JavaScript Funktion die jeweils ein DIV mit
Text füllen:
Beispiel 1
'''javascript
// JavaScript Funktion, die ein DIV mit einer ID findet und Text hinzufügt
6 function insertTextInDiv() {
    let divElement = document.getElementById('myDiv');
    divElement.innerText = 'Hallo Welt!';
}
'''
"""
```

```
'''
11 Beispiel 2
'''javascript
// JavaScript Funktion, die das erste DIV-Element auf der Seite sucht und
  Text einfügt
function insertTextInFirstDiv() {
  let divElement = document.querySelector('div');
16  divElement.innerText = 'Willkommen auf der Webseite!';
}

'''
Aufgabe: "Generiere eine JavaScript-Funktion, die ein DIV-Tag mit einer
  bestimmten Klasse sucht und einen Text einfügt."
21 '''
```

Listing 2.3: Few-Shot Prompt als Python-String

Das Ergebnis vom Modell *deepseek-coder-v1:latest* ist in Listing 2.4 zu sehen.

```
Hier ist eine JavaScript Funktion, die ein DIV-Tag mit einer bestimmten
  Klasse sucht und einen Text hinzufügt:

'''javascript
4 // JavaScript Funktion, die ein DIV mit einer bestimmten Klasse findet und
  Text hinzufügt
function insertTextInDivWithClass(className) {
  let divElement = document.querySelector('div.' + className);
  if (divElement) {
    divElement.innerText = 'Hallo von der Funktion!';
9  } else {
    console.log('Es wurde kein DIV mit der Klasse "' + className + '"
      gefunden.');
```

```
    }
  }
}
'''
14 Du kannst diese Funktion aufrufen und die gewünschte Klasse als Argument ü
  bergeben:

'''javascript
insertTextInDivWithClass('my-class'); // Ersetze 'my-class' durch deine
```

```
19   tatsächliche Klasse  
    '''
```

Listing 2.4: Antwort des Few-Shot-Prompts

Wie erfolgreich diese Technik ist, wird in [5] beschrieben. Wie wichtig bei der Formulierung der Anweisungen das Format und die Beschriftung ist, zeigt [6] in seiner Studie. Wird ein Beispiel angegeben, kann dazu kommen, dass das Modell nicht die richtige Antwort findet. Dann sollten mehrere Beispiele an das Modell übergeben werden.

Chain-of-Thought Prompting (CoT)

Wenn mit den Zero-shot und Few-shot Techniken nicht das gewünschte Ergebnis von den Modellen erzielt wird, könnte die Chain-of-Thought CoT Technik Verwendung finden. Bei dieser Technik wird das Modell aufgefordert, sein Vorgehen zu belegen. Mit dieser Technik kann besser nachvollzogen werden, wie im Modell der Lösungsversuch abläuft.

```
1 prompt_string: str = """  
Generiere eine JavaScript-Funktion, die ein DIV-Tag sucht und einen Text  
  einfügt. Denke Schritt für Schritt darüber nach, wie du die Aufgabe  
  angehst.  
  """
```

Listing 2.5: CoT Prompt als Python-String

Die Antwort.

```
2 Um eine JavaScript-Funktion zu erstellen, die ein DIV-Tag sucht und einen  
  Text einfügt, können wir folgendermaßen vorgehen:  
  
1. **Auswahl des DOM-Elements**: Wir müssen zunächst das DIV-Element  
  finden, dem der Text hinzugefügt werden soll. Hierfür verwenden wir die  
  Methode 'getElementById' oder andere Selektoren wie 'querySelector'.  
  
2. **Einfügen des Textes**: Sobald wir das DIV-Element gefunden haben, kö  
  nnen wir den gewünschten Text in dieses Element einfügen. Dies  
  erreichen wir mithilfe der Eigenschaft 'innerText' oder 'textContent'.  
  
7 Hier ist eine detaillierte Implementierung der Funktion:  
  
  '''javascript
```

```
function insertTextIntoDiv(divId, text) {
    // 1. Auswahl des DOM-Elements
12    var divElement = document.getElementById(divId);

    // Überprüfen, ob das Element existiert
    if (divElement) {
        // 2. Einfügen des Textes
17        divElement.innerText = text;
    } else {
        console.error("Das DIV-Element mit der ID " + divId + " wurde
            nicht gefunden.");
    }
}
22 '''

### Erklärung:
1. **getElementById**: Diese Methode gibt das Element mit der angegebenen
    ID zurück. Wenn das Element nicht existiert, wird 'null' zurückgegeben.
    Daher prüfen wir im Code, ob 'divElement' nicht 'null' ist, bevor wir
    versuchen, den Text einzufügen.
2. **innerText**: Diese Eigenschaft setzt den Text innerhalb des Elements.
    Wenn das Element bereits Inhalt enthält, wird dieser überschrieben.
27

### Verwendung:
Um die Funktion zu verwenden, rufen wir sie mit der ID des DIV-Elements
und dem gewünschten Text auf:

'''html
32 <!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="UTF-8">
    <title>Beispiel</title>
37 </head>
<body>
    <div id="myDiv"></div>
    <script>
        // Funktion aufrufen
42        insertTextIntoDiv('myDiv', 'Hallo, Welt!');
```

```
</script>
</body>
</html>
'''
```

47

In diesem Beispiel wird das DIV-Element mit der ID 'myDiv' gesucht und der Text "Hallo, Welt!" darin eingefügt.

Listing 2.6: Antwort des CoT-Prompts

Wie gut die Technik bei Modellen funktioniert, wird in [7] untersucht.

Meta Prompting

Für Meta Prompting, sind nach [8] die wichtigsten Merkmale wie folgt,

- *Strukturierung* der Prompts beispielsweise bestimmen der Denkweise oder Reihenfolge vorgeben.
- *Syntax fokussiert*, dadurch wird die Syntax als Leitvorlage für die erwartete Antwort verwendet.
- *abstrakte Beispiele* die sich mit der Struktur des Prompt befassen, nicht mit der expliziten Lösung, wie die inhaltsgesteuert Few-Shot-Prompts.
- *vielseitig* lässt es zu, das der Prompt in vielen Bereichen anwendbar ist und geben Antwort auf eine Vielzahl von Problemen, sodass der Prompt nicht jedes Mal neu geschrieben werden muss.
- *kategorischer Ansatz* strukturiert den Aufbau der Prompts in logische Anordnung und Kategorisierung

Dem Modell wird wie bei der CoT Technik angewiesen, sein Vorgehen offen zulegen. Neben dem Ergebnis, wie beim CoT, soll hierbei auch der Ablauf und die Planung der Ergebnisfindung dargestellt werden. Ein Beispiel einer Anweisung könnte folgendermaßen aussehen und wird in Listing 2.7 gezeigt.

```
prompt_string: str = """
2 Bevor du den Code für die Erstellung einer PHP-Funktion schreibst, die
  HTML in ein PDF speichert, erkläre die Wahl der Tools, die notwendigen
  Installationsschritte und wie du die Funktion implementierst. Verwende
  DomPDF und beschreibe den gesamten Prozess in klaren Schritten, bevor
  du den Code präsentierst.
  """
```

Listing 2.7: Meta Prompt als Python-String

Das Listing 2.8 zeigt die ausführliche Antwort. Ebenfalls wurde hier das *deepseek-code-v2:latest* Modell angewandt.

```
Natürlich! Bevor ich dir den Code für eine PHP-Funktion zur Erstellung
    eines HTML-Inhalts zu PDF mit DomPDF schreibe, möchte ich zunächst die
    Wahl der Tools und deren notwendigen Installationsschritte erklären.
    Dann werde ich den gesamten Prozess in klaren Schritten beschreiben.
2
### Wahl der Tools: DomPDF
DomPDF ist eine leistungsstarke PHP-Bibliothek, die es ermöglicht, HTML-
    Inhalte direkt in PDF-Dateien zu konvertieren. Es basiert auf dem
    ebenfalls beliebten TCPDF und bietet einige Vorteile wie bequeme
    Integration und eine benutzerfreundliche API.

### Notwendige Installationsschritte:
7 1. **Serverkonfiguration**: Stellen Sie sicher, dass Ihr Server die
    notwendigen Erweiterungen (z.B. 'php-dompdf') aktiviert hat. Dies kann
    über die '.htaccess'-Datei oder eine Konfigurationsdatei in PHP
    erfolgen.
2. **Bibliotheken herunterladen**: Laden Sie die DomPDF-Bibliothek von der
    offiziellen Website oder einem gängigen Paketmanager wie Composer
    herunter.
3. **Composer verwenden**: Wenn du Composer verwendest, füge dies in
    deiner 'composer.json' Datei hinzu:
12  '''json
    {
        "require": {
            "dompdf/dompdf": "^1.0"
        }
    }
    '''
17 Dann führe 'composer install' aus.
4. **Autoloader einbinden**: In deiner PHP-Datei, die DomPDF verwendet,
    binde den Autoloader von Composer ein:
    '''php
    require_once 'vendor/autoload.php';
    '''
22
### Implementierung der Funktion:
1. **Initialisieren Sie DomPDF**: Erstellen Sie eine neue Instanz von
```

DomPDF in Ihrer PHP-Datei.

2. ****HTML-Inhalt bereitstellen****: Übergeben Sie den HTML-Code oder die URL des HTML-Dokuments an die DomPDF-Instanz.
3. ****PDF generieren****: Rufen Sie die Methode auf, um das PDF zu erzeugen.
- 27 4. ****Speichern oder Anzeigen des PDFs****: Speichern Sie das PDF in einer Datei oder geben Sie es direkt an den Browser aus.

Hier ist ein Beispiel für eine einfache PHP-Funktion, die HTML-Inhalt zu einem PDF konvertiert:

```
32 <?php
require_once 'vendor/autoload.php';

use Dompdf\Dompdf;

37 function generatePDF($html) {
    // Instanz von Dompdf erstellen
    $dompdf = new Dompdf();

    // HTML-Inhalt setzen
42 $dompdf->loadHtml($html);

    // (Optional) Konfigurieren Sie Dompdf, um die Größe des Blattes und
    // andere Einstellungen anzupassen
    $dompdf->setPaper('A4', 'portrait');

47 // Rendern Sie das PDF
    $dompdf->render();

    // Speichern oder Ausgeben des PDFs
    $output = $dompdf->output();
52 file_put_contents('example.pdf', $output);
}

// Beispiel-HTML-Inhalt
$html = '<h1>Hallo, Welt!</h1><p>Dies ist ein Test für die Konvertierung
    von HTML nach PDF.</p>';
57 generatePDF($html);
```



```
?>
'''

### Zusammenfassung:
62 1. **Installiere DomPDF und andere benötigte Bibliotheken**.
   2. **Initialisiere eine Instanz von DomPDF** in deiner PHP-Datei.
   3. **Setze den HTML-Inhalt oder die URL des HTML-Dokuments**.
   4. **Rendern Sie das PDF** und speichern oder geben Sie es aus.
67 Mit diesem Prozess kannst du einfach HTML-Inhalte in professionelle PDFs
   umwandeln, die dann gespeichert oder weiter verwendet werden können.
```

Listing 2.8: Antwort des Meta-Prompts

Meta-Prompts sind Token-Effizient und verringern die benötigte Anzahl an Token, da der Schwerpunkt wie beschrieben auf der Struktur liegt, nicht auf den expliziten Inhalt.

Prompt Chaining

Hierbei wird eine komplexe Aufgabe in Unteraufgaben zerlegt. Die Antwort einer Unteraufgabe dient als Eingabe für die nächste Unteraufgabe. Diese Zerlegung ist hilfreich, um Komplexität einer Aufgabe zu verringern und eine Überforderung der Modelle zu verhindern. Durch diese Technik ist eine schrittweise Näherung an die Gesamtlösung der Aufgabe möglich.

Im Beispiel soll das Sprachmodell wieder eine PHP Funktion schreiben, die eine HTML Zeichenkette als PDF speichert.

```
prompt_string: str = """
Nenne mögliche PHP Bibliotheken, mit der PDF Dateien erstellt werden kö
    nnen.
3  """
```

Listing 2.9: Chain Prompt Nr. 1 als Python-String

```
Es gibt mehrere PHP Bibliotheken, die es ermöglichen, PDF-Dateien zu
    erstellen. Hier sind einige beliebte und weit verbreitete Bibliotheken
    :
2
1. **TCPDF**:
   - TCPDF ist eine leistungsstarke Open-Source-Bibliothek zur Erstellung
     von PDF-Dokumenten in PHP. Es bietet umfassende Unterstützung für
     verschiedene Funktionen wie Texte, Grafiken, Tabellen und mehr.
```

```
7  - [Offizielle Website](https://tcpdf.org/)
   - [GitHub-Repository](https://github.com/tecnickcom/tc-lib-pdf)

12 2. **MPDF**:
   - MPDF ist eine weitere leistungsstarke PHP Bibliothek zur PDF-
     Generierung, die sehr beliebt ist und viele Funktionen bietet,
     einschließlich der Unterstützung für Unicode-Zeichen.
   - [Offizielle Website](https://mpdf.github.io/)
   - [GitHub-Repository](https://github.com/mpdf/mpdf)

17 3. **FPDF**:
   - FPDF ist eine einfache und leichtgewichtige Bibliothek zur PDF-
     Generierung in PHP, die auf der FPDF-Bibliothek basiert. Es ermö-
     glicht das Erstellen von PDF-Dokumenten mit Text, Grafiken und
     vielen anderen Elementen.
   - [Offizielle Website](https://tcpdf.org/downloads/)
   - [GitHub-Repository](https://github.com/fpdf/fpdf)

22 4. **DomPDF**:
   - DomPDF ist eine PHP-Implementierung von PDFLib, einer
     leistungsstarken Bibliothek zur Erstellung von PDF-Dokumenten in C
     ++. Es bietet auch die Möglichkeit, HTML in PDF umzuwandeln.
   - [Offizielle Website](https://dompdf.github.io/)
   - [GitHub-Repository](https://github.com/dompdf/dompdf)

27 5. **SnappyPdf**:
   - SnappyPdf ist eine PHP Bibliothek, die auf der wunderbaren Pdfcrowd
     API basiert und ermöglicht das Erstellen von PDF-Dokumenten aus HTML
     oder Textinhalten.
   - [Offizielle Website](https://github.com/Knplabs/snappy)
   - [GitHub-Repository](https://github.com/Knplabs/snappy)

Diese Bibliotheken bieten verschiedene Funktionen und Flexibilität, sodass
Sie je nach Anwendungsfall die passende Wahl treffen können.
```

Listing 2.10: Antwort des Chain-1-Prompts

```
2 prompt_string: str = ""
  Welche der folgenden Bibliotheken bieten die beste Unterstützung der
  Entwickler,
```

```
7 * TCPDF
  * MPDF
  * FPDF
  * DomPDF
  * SnappyPdf
  """
```

Listing 2.11: Chain Prompt Nr. 2 als Python-String

```
2 Um zu beurteilen, welche der angegebenen Bibliotheken die beste Unterstüt-
  zung für Entwickler bietet, ist es wichtig, verschiedene Kriterien zu
  berücksichtigen. Diese können unter anderem die Funktionalität, die
  Aktualität der Dokumentation, das Umfeld der Community, die Unterstüt-
  zung von Drittanbietern und die allgemeine Benutzerfreundlichkeit
  umfassen.

  Hier ist eine kurze Bewertung einiger dieser Bibliotheken:

  1. **TCPDF**: TCPDF (HTML2FPDF) ist eine leistungsstarke PDF-Bibliothek,
  die hauptsächlich in PHP entwickelt wurde und auf HTML2FPDF basiert. Es
  bietet eine Vielzahl von Funktionen für das Erstellen und Bearbeiten
  von PDF-Dokumenten. Die Dokumentation ist recht ausführlich, und es
  gibt eine aktive Community.

  7 2. **MPDF**: MPDF (Multi-Purpose PDF Library) ist ebenfalls in PHP
  geschrieben und bietet ähnliche Funktionen wie TCPDF. Es unterstützt
  HTML sowohl im Text- als auch im Modus, was es für Entwickler attraktiv
  macht, die mit Webtechnologien arbeiten. Die Dokumentation ist gut und
  die Community ist aktiv.

  3. **FPDF**: FPDF (Free Pascal PDF) ist eine andere weit verbreitete PHP-
  Bibliothek zum Erstellen von PDF-Dokumenten. Es ist in Free Pascal
  geschrieben, aber auch viele PHP-Entwickler bevorzugen es für seine
  einfache Verwendung und die Fähigkeit, HTML zu rendern. Die
  Dokumentation ist gut verständlich.

  4. **DomPDF**: DomPDF ist eine portierte Version von MPDF in PHP, die auf
  der DOM-Basis arbeitet. Es unterstützt das Rendern von HTML und bietet
  eine hohe Flexibilität bei der Erstellung komplexer PDF-Dokumente. Die
  Dokumentation ist gut, aber die Community ist kleiner als bei anderen
```

- 12 Bibliotheken.
5. ****SnappyPdf****: SnappyPdf verwendet die C-Bibliothek "snappy" von Google für das Komprimieren und Dekomprimieren der PDF-Dateien. Es unterstützt viele verschiedene Backends wie Ghostscript, Qpdf oder MuPDF. Die Verwendung erfordert jedoch evtl. zusätzliche Abhängigkeiten und ist nicht in allen Sprachen verfügbar.
- **Empfehlung****: Wenn Sie eine Bibliothek für PHP-Projekte suchen, bei denen HTML unterstützt wird und Sie eine reaktionsfähige Unterstützung der Entwickler sowie eine aktive Community benötigen, würde ich ****TCPDF**** oder ****MPDF**** empfehlen. Beide Bibliotheken sind gut dokumentiert und haben aktive Communities. FPDF ist ebenfalls ein guter Ersatz, vor allem wegen seiner guten Leistung und der größeren Verbreitung im PHP-Umfeld.
- 17 ****SnappyPdf**** sollte nur dann in Betracht gezogen werden, wenn Sie speziell eine plattformübergreifende Lösung benötigen und bereits über die notwendigen Abhängigkeiten verfügen.

Listing 2.12: Antwort des Chain-2-Prompts

```
prompt_string: str = """
Erstelle eine PHP Funktion, die eine HTML Zeichenkette in einer PDF Datei
speichert. Dafür soll die PHP Bibliothek 'DomPDF' verwendet werden.
3 """
```

Listing 2.13: Chain Prompt Nr. 3 als Python-String

Um eine PHP Funktion zu erstellen, die eine HTML-Zeichenkette in einer PDF-Datei speichert, kann die PHP Bibliothek 'DomPDF' verwendet werden. Hier ist ein Beispiel, wie Sie dies tun können:

2

1. Installieren Sie zuerst die DomPDF Bibliothek über Composer:

```
'''bash
composer require dompdf/dompdf
'''
```

7

2. Erstellen Sie eine PHP Funktion, die die HTML-Zeichenkette enthält und die PDF speichert:

```
‘‘‘php
<?php
12 require_once __DIR__ . '/vendor/autoload.php';

use Dompdf\Dompdf;

function saveHtmlToPdf($html) {
17     // Erstellen Sie eine Instanz von Dompdf
    $dompdf = new Dompdf();

    // Setzen Sie die HTML-Inhalte in Dompdf
    $dompdf->loadHtml($html);
22

    // (Optional) Konfigurieren Sie die PDF Optionen
    $dompdf->setPaper('A4', 'portrait');

    // Rendern Sie das PDF
27    $dompdf->render();

    // Speichern Sie das PDF in einer Datei
    $output = $dompdf->output();
    file_put_contents('output.pdf', $output);
32 }

// Beispiel HTML-Inhalt
$html = '<!DOCTYPE html>
<html lang="de">
37 <head>
    <meta charset="UTF-8">
    <title>Beispielseite</title>
</head>
<body>
42     <h1>Hallo Welt!</h1>
    <p>Dies ist ein Beispiel für die Verwendung von DomPDF in PHP.</p>
</body>
</html>';

47 // Rufen Sie die Funktion auf, um das HTML als PDF zu speichern
```

```
saveHtmlToPdf($html);
?>
'''
```

52 In diesem Beispiel:

1. Wir laden zuerst die DomPDF Bibliothek über Composer.
2. Eine PHP Funktion 'saveHtmlToPdf' wird definiert, die eine HTML-Zeichenkette als Parameter enthält.
3. Innerhalb der Funktion erstellen wir eine Instanz von Dompdf und setzen den in der Funktion übergebenen HTML-Inhalt.
4. Wir rendern das PDF und speichern es als 'output.pdf'.

57

Sie können diese Funktion aufrufen und die gewünschte HTML-Zeichenkette an sie übergeben, um eine PDF-Datei zu erstellen und zu speichern.

Listing 2.14: Antwort des Chain-3-Prompts

Tree of Thoughts (ToT)

Diese Technik wurde von [9] und [10] vorgeschlagen. (ToT) kommt bei komplexen Anforderungen zum Einsatz, wenn einfache Techniken, die zuvor genannt wurden, nicht mehr ausreichen. Auch bei dieser Technik wird die Anforderung in keine Aufgaben zerlegt. Dann werden mehrere Lösungen pro Aufgabe erstellt und im Anschluss bewertet. Dabei entsteht eine Baustruktur, von der die besten Lösungen ausgesucht werden.

Im Folgenden ein Beispiel einer Teilaufgabe, bei der drei mögliche Lösungsvorschläge vom Modell erstellt wurden. Die Listings 2.15, 2.16 und 2.17 zeigen die Ergebnisse des Modells. Es wurden jeweils andere PHP Bibliotheken für die Lösung vorgeschlagen. Als Modell wurde *deepseek-coder-v2* verwendet. Als Nutzereingabe wurde folgendes an das Modell übergeben,

Erstelle drei verschiedene Methoden in PHP, die eine HTML Zeichenkette in ein PDF umwandeln und es als Datei, mit angegebenen Namen speichern.

Die erste Lösung beinhaltet die Bibliothek *Dompdf*.

```
<?php
2 require 'vendor/autoload.php';

use Dompdf\Dompdf;

function createPDF($html, $filename)
```

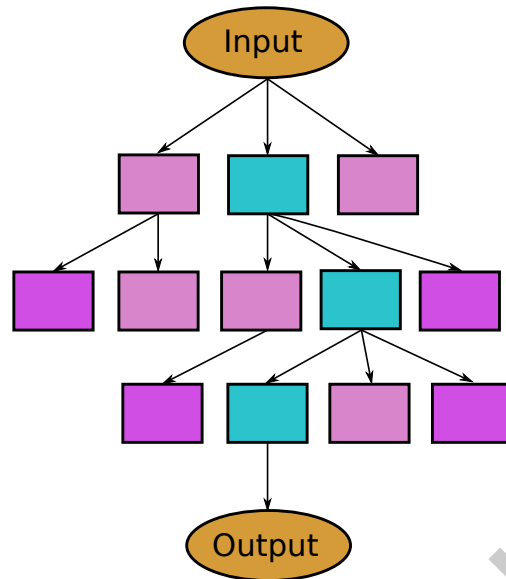


Abbildung 2.4: Baumstruktur der „Tree of Thoughts“ Technik

```

7 {
    $dompdf = new Dompdf();
    $dompdf->loadHtml($html);
    $dompdf->setPaper('A4', 'portrait');
    $dompdf->render();
12 file_put_contents($filename, $dompdf->output());
}

// ... more.

```

Listing 2.15: Ausgabe für DOMPDF Bibliothek

```

<?php
require_once('mpdf/autoload.inc.php');

use Mpdf\Mpdf;
5
function createPDF($html, $filename)
{
    $mpdf = new Mpdf();
    $mpdf->WriteHTML($html);
10 $mpdf->Output($filename, 'F');
}

```

```
}  
  
// ... more.
```

Listing 2.16: Ausgabe für MPDF Bibliothek

Als dritte Ausgabe liefert coder,

```
<?php  
2 require_once('tcpdf/tcpdf.php');  
  
class CustomPDF extends TCPDF  
{  
    public function Header()  
7    {  
        $this->SetFont('helvetica', 'B', 20);  
        $this->Cell(0, 15, 'Beispiel-PDF', 0, false, 'C', 0, '', 0, false, 'M'  
            , 'M');  
    }  
}  
12  
function createPDF($html, $filename) {  
    $pdf = new CustomPDF(PDF_PAGE_ORIENTATION, PDF_UNIT, PDF_PAGE_FORMAT,  
        true, 'UTF-8', false);  
    $pdf->SetDefaultMonospacedFont(PDF_FONT_MONOSPACED);  
    $pdf->SetMargins(PDF_MARGIN_LEFT, PDF_MARGIN_TOP, PDF_MARGIN_RIGHT);  
17 $pdf->SetHeaderMargin(PDF_MARGIN_HEADER);  
    $pdf->SetFooterMargin(PDF_MARGIN_FOOTER);  
    $pdf->setPrintHeader(false);  
    $pdf->setPrintFooter(false);  
    $pdf->AddPage();  
22 $pdf->writeHTML($html, true, false, true, false, '');  
    $pdf->Output($filename, 'F');  
}  
  
// ... more.
```

Listing 2.17: Ausgabe für TCPDF Bibliothek

2.5.2 Grenzen beim Prompt-Engineering für LLMs

Trotz der bemerkenswerten linguistischen Leistung, stoßen große Sprachmodelle an ihre Grenzen, unter anderem wie in [3] beschrieben,

2.6 Grundlagen bei der Entwicklung von Webanwendungen

Webanwendung

Entwurf

STAND DER FORSCHUNG

3.1 Methoden und Ansätze

3.2 Forschungslücken und zukünftige Forschung

3.2.1 Identifikation von Forschungslücken

3.2.2 Zukünftige Forschungsrichtungen

Entwurf

METHODIK

4.1 Auswahl der LLM

4.2 Prompt-Engineering

Entwurf

Entwurf

IMPLEMENTIERUNG

5.1 Modelle lokal aufsetzen

Als Server dient ein Debian 12 System.

5.1.1 Install Ollama

Ein Skript ausführen

```
curl -fsSL https://ollama.com/install.sh | sh
```

Ein Model laden und im Anschluss starten, Beispiel

```
ollama pull deepseek-coder-v2:16b \
```

```
ollama run deepseek-coder-v2:16b
```

Config Ollama

Set correct IP and Port in /etc/systemd/system/ollama.service

```

4 diff --git a/ollama.service b/ollama.service
  --- a/ollama.service
  +++ b/ollama.service
  @@ -10,3 +10,4 @@
  RestartSec=3
  Environment="PATH=/usr/local/bin:/usr/bin"
  -
  + Environment="OLLAMA_HOST=0.0.0.0"
  +
  9
```

Listing 5.1: Ollama Hostanpassng für Netzwerkbetrieb

5.1.2 Open WebUI

Optional kann ein grafisches Tool, zum Testen und verwalten vom Ollama-Server im Netzwerk installiert werden. Der Aufruf der UI, kann mittel Browser erfolgen. Hier wird die IP und der Port 8080 angegeben. Beispiel <http://192.168.2.45:8080>.

```
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
-o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] \
https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io \
docker-buildx-plugin docker-compose-plugin
docker run -d --network=host -v open-webui:/app/backend/data \
-e OLLAMA_BASE_URL=http://127.0.0.1:11434 --name open-webui \
--restart always ghcr.io/open-webui/open-webui:main
```

5.1.3 Python Client

```
pip3 install langchain
pip3 install ollama
pip3 install mistral
```

EVALUATION

6.1 Einfache HTML Seite

6.1.1 ChatGPT 3.5

Entwurf

Entwurf

Entwurf

Entwurf

DISKUSSION UND AUSBLICK

8

Entwurf

Entwurf

FAZIT

9

Entwurf

Entwurf

LITERATUR

- [1] Erin Yepis. *Developers want more, more, more: the 2024 results from Stack Overflow's Annual Developer Survey*. 24. Juli 2024. URL: <https://stackoverflow.blog/2024/07/24/developers-want-more-more-more-the-2024-results-from-stack-overflow-s-annual-developer-survey/> (besucht am 09.08.2024).
- [2] Pekka Ala-Pietilä u. a. *Eine Definition der KI: Wichtigste Fähigkeiten und Wissenschaftsgebiete*. 5. März 2019. URL: https://elektro.at/wp-content/uploads/2019/10/EU_Definition-KI.pdf (besucht am 10.09.2024).
- [3] Xavier Amatriain. *Prompt Design and Engineering: Introduction and Advanced Methods*. 24. Jan. 2024. URL: <https://arxiv.org/abs/2401.14423v3> (besucht am 12.10.2024).
- [4] Jason Wei u. a. *Finetuned language models are Zero-Shot learners*. 3. Sep. 2021. URL: <https://arxiv.org/abs/2109.01652> (besucht am 12.10.2024).
- [5] Tom B. Brown u. a. *Language Models are Few-Shot Learners*. 28. Mai 2020. URL: <https://arxiv.org/abs/2005.14165> (besucht am 12.10.2024).
- [6] Sewon Min u. a. *Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?* 25. Feb. 2022. URL: <https://arxiv.org/abs/2202.12837> (besucht am 12.10.2024).
- [7] Jason Wei u. a. *Chain-of-Thought prompting elicits reasoning in large language models*. 28. Jan. 2022. URL: <https://arxiv.org/abs/2201.11903> (besucht am 12.10.2024).
- [8] Yifan Zhang, Yang Yuan und Andrew Chi-Chih Yao. *Meta Prompting for AI Systems*. 20. Nov. 2023. URL: <https://arxiv.org/abs/2311.11482> (besucht am 12.10.2024).
- [9] Jieyi Long. *Large language model guided Tree-of-Thought*. 15. Mai 2023. URL: <https://arxiv.org/abs/2305.08291> (besucht am 14.10.2024).

- [10] Shunyu Yao u. a. *Tree of Thoughts: Deliberate Problem Solving with Large Language Models*. 17. Mai 2023. URL: <https://arxiv.org/abs/2305.10601> (besucht am 14.10.2024).

Entwurf

ANHANG

Entwurf