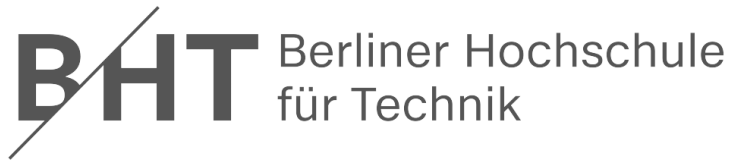
Evaluierung und Optimierung von Large Language Models für die Entwicklung von Webanwendungen

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 14. Oktober 2024

Entwurf

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 14. Oktober 2024

Unterschrift

Entwurf

Abstract

Abstract in Englisch.

Entwurf

Zusammenfassung

Zusammenfassung in Deutsch.

Entwurf

Inhaltsverzeichnis

Entwurf

Entwurf

Abbildungsverzeichnis

Entwurf

Entwurf

Tabellenverzeichnis

Entwurf

Entwurf

Listings

Entwurf

Entwurf

Abkürzungsverzeichnis

k-NN k-Nearest Neighbors.

KI Künstliche Intelligenz.

LLM Large Language Model.

ML Maschine Learning.

Entwurf

Einleitung

1

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

Entwurf

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.*
>
> Vattenfall Online , KI für Unternehmen – die Grenzen der KI

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.

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 verwendetet 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.

Entwurf

1.5 Abgrenzung

Ausschluss anderer Anwendungsbereiche.

Rechtliche und ethische Überlegungen werden nur am Rande berücksichtigt.

Entwurf

Entwurf

Grundlagen

2

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.

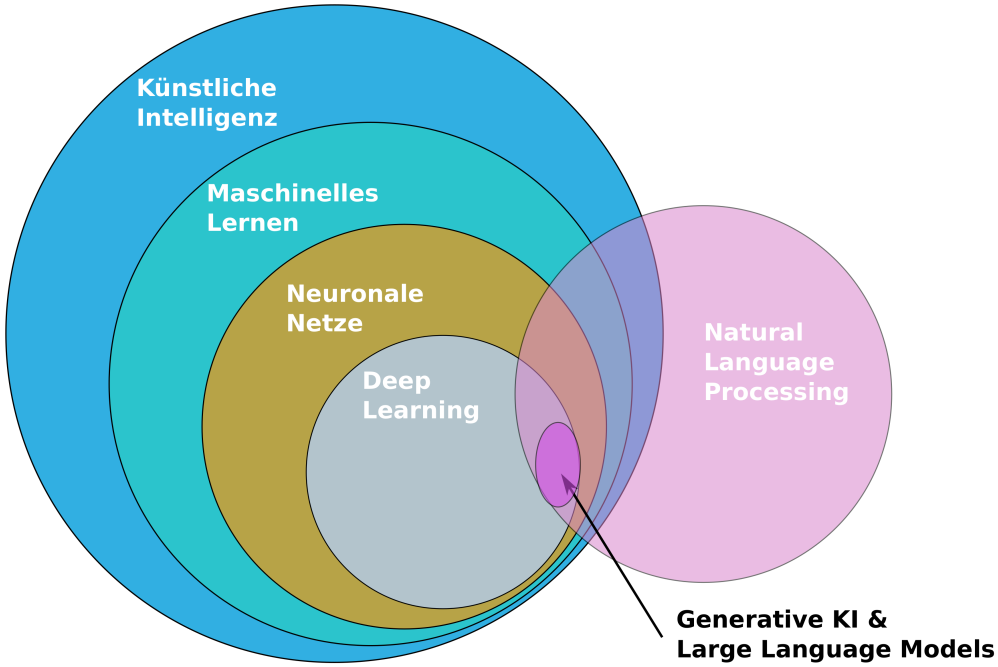


Abbildung 2.1: Einordnung LLMs

Nurf

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

Entwurf

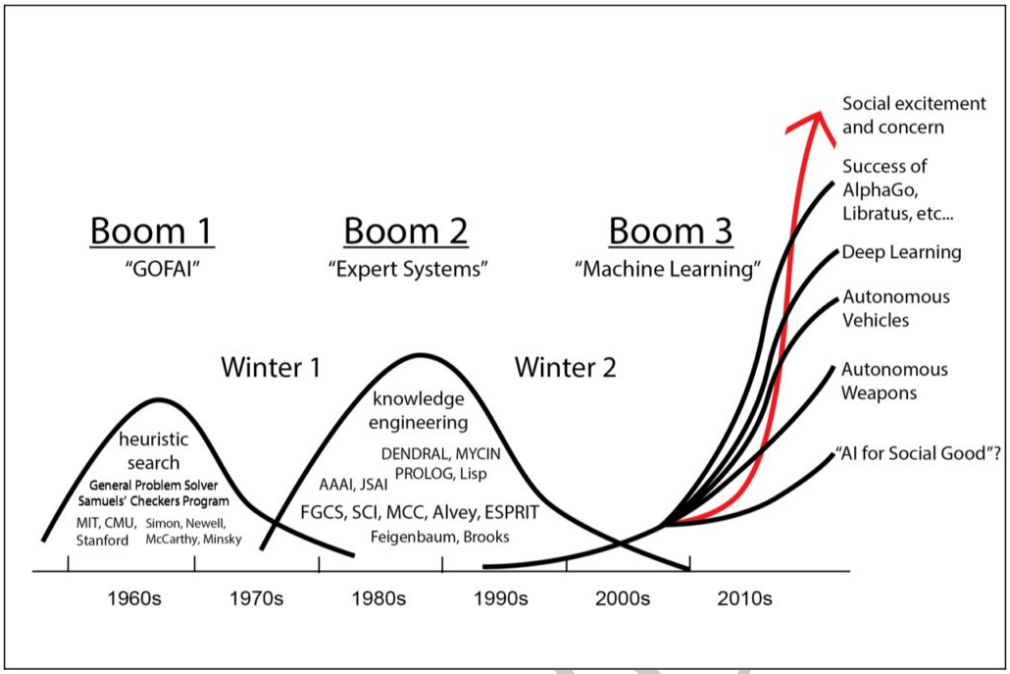


Abbildung 2.2: KI Winterzyklen

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 verwendet.

> *A computer programm is said to learn from experience E with respekt 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,

Entwu

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.

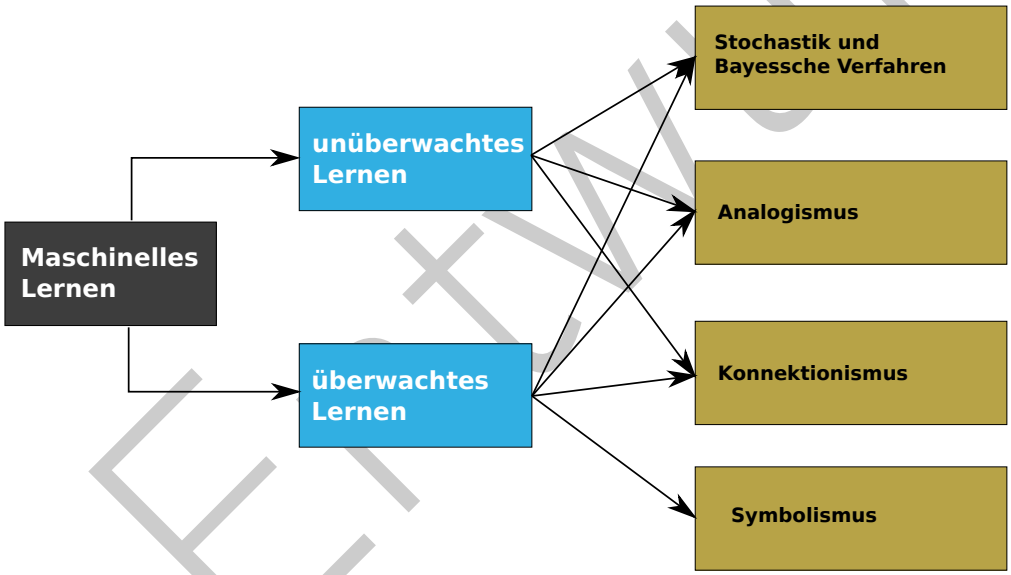


Abbildung 2.3: Lernparadigmen des maschinellen Lernens

Entwurf

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 überwachten 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 unüberwachtes 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 vergleicht und nach ähnlichen Strukturen und Mustern sucht. 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 kleine Einheiten die miteinander verbunden sind. Diese werden als Neuronen bezeichnet, die den Nervenzellen von Organismen nachempfunden sind. Die

Entwurf

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

Entwurf

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

Entwurf

2.5 Prompt Engineering

Prompt Engineering optimiert die Antworten große Sprachmodelle, ohne Parameter, wie Bias und Gewichte des Models 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 diesen Anweisungen handelt es sich um einfache Anweisungen, ohne Angabe von Beispielen und sonstigen zusätzlichen Informationen.

[Beispiel Zero-shot]

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.

[Beispiel Few-shot]

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.

Entwurf

Chain-of-Thought Prompting

Wenn mit den Zero-shot und Few-shot Techniken nicht das gewünschte Ergebnis von den Modellen erzielt wird, könnte die Chain-of-Thought Technik Verwendung finden. Bei dieser Technik wird das Modell aufgefordert, sein Vorgehen zu belegen.

[Beispiel chain of thought]

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

[Beispiel für Meta-Prompt]

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.

Tree of Thoughts

Diese Technik wurde von [9] und [10] vorgeschlagen.

Entwurf

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

```php
<?php
// Methode 1: Verwendung von dompdf/dompdf
function htmlToPdfDompdf($html, $filename) {
  require_once 'vendor/autoload.php';
  use Dompdf\Dompdf;

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

// Beispielaufruf
$html = "<h1>Hello World</h1><p>This is a test PDF.</p>";
$filename = "output.pdf";
htmlToPdfDompdf($html, $filename);
```

Listing 2.1: Ausgabe für DOMPDF Bibliothek

```php
<?php
// Methode 2: Verwendung von mpdf/mpdf
function htmlToPdfMpdf($html, $filename) {
  require_once 'vendor/autoload.php';
  use Mpdf\Mpdf;

  $mpdf = new Mpdf();
  $mpdf->WriteHTML($html);
  file_put_contents($filename, $mpdf->Output('', 'S'));
}

// Beispielaufruf
$html = "<h1>Hello World</h1><p>This is a test PDF.</p>";
$filename = "output.pdf";
htmlToPdfMpdf($html, $filename);
```

Listing 2.2: Ausgabe für MPDF Bibliothek

Als dritte Ausgabe liefert coder,

```php
<?php
// Methode 3: Verwendung von TCPDF
function htmlToPdfTcpdf($html, $filename) {
  require_once 'vendor/autoload.php';
  use TCPDF;

  $pdf = new TCPDF(
    PDF_PAGE_ORIENTATION,
    PDF_UNIT,
    PDF_PAGE_FORMAT,
    true,
    'UTF-8',
    false
  );
  $pdf->AddPage();
  $pdf->writeHTML($html, true, false, true, false, '');
  file_put_contents($filename, $pdf->Output('', 'S'));
}

// Beispielaufruf
$html = "<h1>Hello World</h1><p>This is a test PDF.</p>";
$filename = "output.pdf";
htmlToPdfTcpdf($html, $filename);
```

Listing 2.3: 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

3.1 Methoden und Ansätze

3.2 Forschungslücken und zukünftige Forschung

3.2.1 Identifikation von Forschungslücken

3.2.2 Zukünftige Forschungsrichtungen

Entwurf

Entwurf

Methodik

4

4.1 Auswahl der LLM

4.2 Prompt-Engineering

Entwurf

Entwurf

Implementierung

5

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 Post in /etc/systemd/system/ollama.service

```
  diff --git a/ollama.service b/ollama.service
2 --- a/ollama.service
  +++ b/ollama.service
  @@ -10,3 +10,4 @@
  RestartSec=3
  Environment="PATH=/usr/local/bin:/usr/bin"
7 -
  +    Environment="OLLAMA_HOST=0.0.0.0"
  +
```

Listing 5.1: Ollama Hostanpasssng für Netzwerkbetrieb

Entwurf

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
```

Entwurf

Evaluation

6

6.1 Einfache HTML Seite

6.1.1 ChatGPT 3.5

Entwurf

Entwurf

Anwendungsszenarien

7

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).

Entwurf

[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