Studiengang: Data Science & Management (M.Sc.)

Betreuer: Prof. Dr. Marcel Hebing

**Der Einfluss von Prompt Engineering auf Large Language Models im Argument Mining**

Masterarbeit

zur Erlangung des akademischen Grades eines

Masters of Science (M.Sc.)

Eingereicht von Benjamin Fels

Matrikelnummer: 190200

Datum: 28.01.2024

Executive Summary

Eine vollständige Zusammenfassung mit Fokus aus actionable insights

Inhaltsverzeichnis

[1 Einleitung 1](#_Toc186791521)

[1.1 Begriffsdefinition + Stand der Forschung 4](#_Toc186791525)

[1.1.1 Argumente und Argument Mining 4](#_Toc186791526)

[1.1.2 Aufgaben des Argument minings 5](#_Toc186791527)

[1.1.3 Argument diagramming techniques / Argument Schemes 12](#_Toc186791528)

[1.1.4 Datensatz / Annotation von Argumenten 12](#_Toc186791529)

[1.1.5 Large Language Models 14](#_Toc186791530)

[2 Daten und Methoden 18](#_Toc186791531)

[2.1 Modell / LLM / Large Language Model 18](#_Toc186791532)

[2.2 Datensatzbeschreibung 19](#_Toc186791533)

[2.3 Methode 22](#_Toc186791534)

[2.3.1 Tech Stack 22](#_Toc186791535)

[2.3.2 Metriken zur Evaluation 22](#_Toc186791536)

[2.3.3 Eingabeaufforderungen / Prompts 25](#_Toc186791537)

[3 Ergebnisse 30](#_Toc186791552)

[4 Diskussion und Handlungsempfehlungen 30](#_Toc186791553)

[5 Fragestellung 32](#_Toc186791554)

[6 Literaturverzeichnis / Quellenverzeichnis 33](#_Toc186791555)

[7 Anhänge & Projektdokumentation 37](#_Toc186791556)

[7.1 Modellvergleich 37](#_Toc186791557)

[7.2 Prompt Templates 39](#_Toc186791558)

# Einleitung

- Vorarbeiten (eigene und extern, v.a. Literaturrecherche)

- Kontext der Arbeit klar verständlich machen

- Entwicklung der Forschungsfrage(n)

- Einleitung nicht zu kurz machen, mach immerhin 25% der Arbeit aus

In einer Zeit, in der Informationen in großer Menge und hoher Geschwindigkeit verbreitet werden, ist es von wachsender Bedeutung, wesentliche Aussagen schnell und präzise zu erfassen. Künstliche Intelligenz kann den Menschen bei seinen täglichen Aufgaben unterstützen.

Argumente sind ein wichtiger Bestandteil in der menschlichen Kommunikation. Peldszus & Stede (2013, S. 1) bezeichnen Argumentationen sogar als einen der zentralen Aspekte der menschlichen Kommunikation. Dabei werden Standpunkte anhand von Beispielen bestärkt mit dem Ziel die andere Seite von dem eigenen Standpunkt zu überzeugen. Gute Argumente sind zudem die Grundlage für eine fundierte Entscheidungsfindung bei verschiedenen Standpunkten (Stab & Gurevych, 2014, S. 1501). Das verstehen der argumentativen Struktur macht es nachvollziehbar, warum Menschen eine gewisse Meinung zu einem Thema haben (Cabrio & Villata, 2018, S. 5428; Lawrence & Reed, 2020, S. 765)

Stab & Gurevych (2014, S. 1501) führen an, dass die automatisierte Erkennung von Argumenten in Texten dazu beitragen kann, die Plausibilität der Argumentationsführung zu prüfen. Der Bereich, welcher sich mit der automatisierten Identifikation von Argumenten in Texten und der Beziehungen der Argumentationskomponenten untereinander auseinandersetzt, nennt sich Argument Mining. Klassische Ansätze für Argument Mining setzen häufig auf umfangreiche manuelle Identifikation, regelbasierte Verfahren oder spezialisierte maschinelle Lernmodelle, die auf spezifische Datensätze trainiert werden (Lawrence & Reed, 2020; Stab & Gurevych, 2017b). Diese Ansätze sind oft zeit- und kostenintensiv und erfordern Fachexpertise. Große Sprachmodelle (engl. Large Language Models, kurz LLMs) ermöglichen hierfür neue Ansätze. Sie liefern dem Stand der Technik entsprechende Ergebnisse bei gängigen Natural Language Processing (NLP) Aufgaben (Ozdemir, 2024, S. 46; Patil & Gudivada, 2024, S. 1). Zu diesen NLP-Aufgaben gehören beispielsweise maschinelle Übersetzung, Beantwortung von Fragen und Textgenerierung (Kochmar, 2022). Sprachmodelle sind nach Patil & Gudivada (2024, S. 4) Modelle, welche die Abfolge von Buchstaben oder Wörtern in Texten vorhersagen. LLMs sind demnach eine spezielle Form, welche auf einem großen Textkorpus trainiert wurden. Sie besitzen eine wesentlich größere Anzahl an Parametern und können eine Vielzahl von sprachbasierten Aufgaben durchführen. Bekannte LLMs sind die Generative Pre-trained Transformer (GPT) von OpenAI.

Hier die Inhalte einfügenwarum das Trainieren eines eigenen Modells sehr aufwändig ist.

Zur Orientierung, für das Training des Llama 3.2-3B instruct / GPT 4o-mini Models wurden x …Grafikkarten verwendet. Eine einzelne solcher Grafikkarten kostet (Stand 12/2024) … €. Diese Grafikkarte wurden über einen Zeitraum von insgesamt … Stunden/Wochen verwendet. Die Kosten für den Strom und das Fachpersonal sind zusätzlich zu berücksichtigen.

(Brown et al., 2020, S. 9): V100 GPU´s, see Appendix B

Obwohl die Architektur der LLMs aufgabenunabhängig konzipiert ist, wird für ein Fine-Tuning ein aufgabenspezifischer Datensatz in einem Umfang von mehreren tausend bis hunderttausend Fällen benötigt (Brown et al., 2020, S. 3). Dies ist nicht bei jedem Anwendungsfall möglich. Zudem müsste für jede spezifische Aufgabe ein eigenes Modell trainiert werden, wofür wiederum jeweils eigene ausreichend große Datensätze benötigt werden würden (Brown et al., 2020, S. 6; Patil & Gudivada, 2024, S. 18).

Leistung von LLMs, die für „jederman“ verfügbar sind bei komplexen Aufgaben.

Die Ressourcen hierzu stehen nicht jedem zur Verfügung. Eine Alternative stellt das Prompt Engineering bzw. Prompting dar, welches sich auf die Methode/Prozess bezieht, einem LLM präzise Eingabeaufforderungen (engl. prompt) zu übergeben, um eine gewünschte Antwort des LLMs zu erhalten (Maharjan et al., 2024, S. 8; Trad & Chehab, 2024, S. 371).

„Prompt engineering is typically the best place to start\*\*. It is often the only method needed for use cases like summarization, translation, and code generation where a zero-shot approach can reach production levels of accuracy and consistency.“ (OpenAI, 2024d)

In der geplanten Untersuchung soll die Wirksamkeit dieser Methoden an dem Anwendungsfall des Argument Minings untersucht werden. Der Einsatz vortrainierter Modelle ohne umfangreiche Anpassung könnte den Aufwand für Argument Mining erheblich reduzieren. Trotz der Popularität großer Sprachmodelle gibt es bislang wenig systematische Studien, die sich explizit auf deren Anwendung für Argument Mining und die Rolle von Eingabeaufforderungen fokussieren. Dort soll die geplante Untersuchung ansetzen.

Die Masterarbeit zielt darauf ab, die Potenziale und Herausforderungen großer Sprachmodelle im Kontext des Argument Minings zu erforschen. Der Fokus liegt dabei auf der Anwendung von Prompt Engineering, um die Generalisierungsfähigkeiten dieser Modelle gezielt zu steuern und deren Leistung ohne Fine-Tuning zu maximieren. Die zentrale **Forschungsfrage** lautet: Wie beeinflusst der Einsatz von Prompt Engineering Techniken die Leistung von Large Language Models bei der automatisierten Erkennung von Argumentationskomponenten und deren Strukturen?

Ansatz zum automatisieren des Prozesses, zum Überführen von unstrukturierten daten (Text) in strukturierte Argumente.

Zur Beantwortung der Forschungsfrage bedarf es (Aufbau der Untersuchung beschreiben) Begriffsdefinition, Stand der Forschung

## Begriffsdefinition + Stand der Forschung

Definition von wesentlichen Begriffen aus der Forschungsfrage:

* Prompt Engineering & Prompt Engineering Techniken. Die Eingabeaufforderungen werden systematisch variiert und deren Leistung untersucht.
* Large Language Models
* Argument Mining / Argumentationskomponenten

### Argumente und Argument Mining

Nach Peldszus & Stede (2013) sowie Stab & Gurevych (2017b) besteht ein Argument aus mehreren Komponenten wie Behauptungen und Prämissen, welche eine bestimmte Struktur durch die Beziehungen zwischen ihnen aufweisen. Demnach wird unter einer Behauptung eine kontroverse Aussage verstanden, welche den zentralen Bestandteil eines Arguments darstellt. Prämissen sind hingegen Gründe für die Rechtfertigung oder Widerlegung solch einer Behauptung. Unter Argument Mining kann hingegen im Hinblick auf diverse Definitionen (Cabrio & Villata, 2018, S. 5427; Lawrence & Reed, 2020, S. 766; Peldszus & Stede, 2013, S. 2; Yeginbergen et al., 2024, S. 11688) die automatische Identifikation und Extraktion der Argumentationskomponenten und deren Beziehungen zueinander aus Texten verstanden werden. Argument Mining stammt aus dem Bereich des NLP (Yeginbergen et al., 2024, S. 11687), welcher wiederum ein Teil aus dem Bereich der künstlichen Intelligenz ist (Kochmar, 2022; Lu et al., 2024, S. 2). Hier ggf. künstliche Intelligenz definieren

Lawrence & Reed (2020) geben in Ihrer Arbeit eine Übersicht über den Forschungsstand zum Thema Argument Mining. *Der Schwerpunkt liegt hierbei auf verschiedenen Ebenen der argumentativen Komplexität. „Es werden diverse Techniken, einschließlich statistischer und linguistischer Methoden durchgeführt. Wir haben eine Hierarchie von Aufgabentypen vorgestellt, die auf zunehmender argumentativer Komplexität basiert.“*

### Aufgaben des Argument minings

Argument Mining lässt sich wiederum in Teilaufgaben zerlegen. Auch hier gibt es in der Literatur abweichende Ansichten, wie diese Teilaufgaben zu unterteilen sind. Es werden sowohl zwei (Cabrio & Villata, 2018, S. 5428; Yeginbergen et al., 2024, S. 11687) als auch drei (Lawrence & Reed, 2020, S. 787–788; Stab & Gurevych, 2017b, S. 620–621) Teilaufgaben benannt. Inhaltlich sind sich die Teilaufgaben sehr ähnlich und werden je nach Vorgehensweise zusammengefasst. Für diese Untersuchung wird die folgende dreiteilige Gliederung der Teilaufgaben herangezogen. Zunächst wird der argumentative Text von dem nicht-argumentativen Text getrennt, gefolgt von der Unterteilung der Argumentationskomponenten in Behauptungen und Prämissen. Abschließend werden die argumentativen Beziehungen identifiziert.

Da die Teilaufgaben aufeinander aufbauen, wirken sich Fehler am Anfang negativ auf die nachfolgenden Aufgaben aus (Stab & Gurevych, 2017b, S. 648–649). Anstatt jedoch Modelle für jede einzelne Teilaufgabe zu trainieren wie es beispielsweise in Stab & Gurevych (2017b) erfolgt ist, soll sich die Lernfähigkeit von LLMs zu Nutze gemacht werden.

Die Erkennung von Argumenten in Textdokumenten umfasst daher mehrere Teilaufgaben: (1) die Trennung von argumentativen und nicht-argumentativen Texteinheiten, (2) die Identifizierung von Behauptungen und Prämissen und (3) die Identifizierung von Beziehungen zwischen Argumentkomponenten. (Stab & Gurevych, 2014, S. 1501)

Anhand der argumentativen Beziehungen können Argumentationsgraphen konstruiert werden. Es gibt verschiedene Theorien zu den Strukturen von Argumenten mit verschiedener Komplexität (Peldszus & Stede, 2013, S. 3–14)

. Peldszus & Stede, 2013 führen eine Reihe von Konstellationen der Beziehgungen der Argumentationskomponenten auf.

Neuere Argument Mining Ansätze betrachten die Extraktion der Argumente als eine Sequenzetikettierungsaufgabe (sequence labeling task) (Cheng et al., 2022, S. 2282; Stab & Gurevych, 2017b, S. 636; Yeginbergen et al., 2024, S. 11688). POS-Tagging und NER sind auch sequence labeling tasks

Die manuelle Identifizierung von Argumenten in einem Text ist sehr aufwändig. Bei der Annotation von Argument-Datensätzen werden Richtlinien erstellt, welche von den Annotoren bei der Beschriftung herangezogen werden. Trotz dieser Richtlinien kann es zu Abweichungen aufgrund persönlicher Interpretation kommen. Beispiel/Quelle hier anführen.

### Argument diagramming techniques / Argument Schemes

Nach (Peldszus & Stede, 2013, S. 6) können Argumente mit ihren Komponenten und Beziehungem in einem Argument-Graphen abgebildet werden, mit dem Prämissen und Schlossfolgerungen als Knoten und der Beziehung zwischen Ihnen als Pfeil. Die einfachste Konstellation eines Arguments wäre demnach eine Schlussfolgerung, welche von einer Prämisse unterstützt wird. Auch wenn eine visuelle Darstellung von Argumentationen die Nachvollziehbarkeit unterstützt, ist dies lediglich informativ aufgeführt und nicht Teil der vorliegenden Untersuchung.

### Datensatz / Annotation von Argumenten

*„Die Zuverlässigkeit eines annotierten Korpus wird durch die Berechnung der Übereinstimmung zwischen den Annotatoren gewährleistet, die den Grad der Übereinstimmung bei der Durchführung der Annotationsaufgabe zwischen den beteiligten Annotatoren misst.* *Bei der Erstellung eines Datensatzes für die Vorhersage von Beziehungen ist das statistische Maß für die Berechnung der Übereinstimmung zwischen den von den Kommentatoren vergebenen Bezeichnungen der Cohen's Kappa-Koeffizient. Wenn die Bewerter vollständig übereinstimmen, dann = 1, wenn es keine Übereinstimmung zwischen den Bewertern gibt, die über das hinausgeht, was zufällig zu erwarten wäre, = 0. Bei NLP-Aufgaben wird die Übereinstimmung als signifikant angesehen, wenn sie >0,6 ist.“* (Cabrio & Villata, 2018, S. 5428)🡪 For more details about inter-annotator agreement, we refer the reader to [Artstein, 2017] 🡪 [Artstein, 2017] Ron Artstein. Inter-annotator Agreement, pages 297–313. Springer Netherlands, Dordrecht, 2017.

Einen Vergleich verfügbarer Datensätze bieten beispielsweise (Cabrio & Villata, 2018, S. 5432).

*„Darüber hinaus wurden seit Beginn der Forschung im Bereich AM verschiedene heterogene Datensätze erstellt. Aufgrund der Unausgereiftheit eines aufstrebenden Feldes und des Mangels an klaren Definitionen wurde jeder Datensatz auf der Grundlage leicht unterschiedlicher Definitionen der Argumentkomponenten und der zwischen ihnen bestehenden Beziehungen annotiert, was die Möglichkeit eines einfachen Abgleichs zwischen den Datensätzen verhindert.“* (Cabrio & Villata, 2018, S. 5432).

(Lawrence & Reed, 2020, S. 780)*: „Eine der Herausforderungen, mit denen sich die derzeitigen Ansätze zum Argument Mining konfrontiert sehen, ist der Mangel an großen Mengen an angemessen kommentierten Argumenten, die als Trainings- und Testdaten dienen können. In jüngster Zeit wurden mehrere Anstrengungen unternommen, um diese Situation durch die Erstellung von Korpora in einer Reihe von verschiedenen Bereichen zu verbessern.“*

*„Obwohl das Ziel des Argument Mining die Extraktion von Argumentationsstrukturen aus natürlichem Text ist, stellt die Verfügbarkeit großer Mengen entsprechend annotierter Trainingsdaten eine große Herausforderung bei der Durchführung dar.“* (Lawrence & Reed, 2020, S. 798)Als Alternative Ansatz Verwendung von Prompt Engineering, da wenig Daten verwendet werden.

Um dem Mangel an annotierten Daten entgegenzuwirken, beschäftigen sich neuere Arbeiten mit der Erstellung von Annotations-Richtlinien (Lawrence & Reed, 2020, S. 806). Diese Verwendung der spezifischen Annotations-Richtlinien bedeutet, dass diese sich auf den jeweiligen Bereich in dem sie entwickelt wurden, beschränken(Lawrence & Reed, 2020, S. 806). Einen einheitlichen Methodik gibt es derzeit nicht. (Lawrence & Reed, 2020, S. 786)

In (Cabrio & Villata, 2018, S. 5429) tragen die leistungsfähigsten Algorithmen, die am häufigsten verwendeten Merkmale, und die freigegebenen Datensätze zusammen. Demnach werden syntaktische und positionsbezogene Merkmale am häufigsten verwendet (Cabrio & Villata, 2018, S. 5431)

### Large Language Models

Abgrenzung Prompt Engineering, Fine-Tuning und RAG

der

Einordnung von LLMs in den Kontext KI

Patil & Gudivada (2024, S. 3) unterscheiden die Phasen der transformerbasierten LLMs in *pretraining*, *transfer learning* und *in-context learning*, welche jeweils wiederum unterteilt werden.

Sprachmodelle besitzen zwei Hauptschritte: *pretraining* und *transfer learning* (Patil & Gudivada, 2024, S. 3)*.*

Die *attention layer* hilft dem *decoder* dabei zu verstehen, welcher Input besonders wichtig ist (Patil & Gudivada, 2024, S. 4).

„In general, autoencoding models learn bidirectional contextualized representation suited for NLU tasks, whereas autoregressive models learn to generate the next token and hence are suited for NLG tasks.“ (Patil & Gudivada, 2024, S. 8)

Transformer ermöglichen parallele Verarbeitung, entgegen RNNs welche nur zur sequenziell (nacheinander) Berechnung in der Lage sind, dadurch verringert sich die Trainingszeit (Patil & Gudivada, 2024, S. 6).

Die Integration von domänenspezifischen Wissen in LLMs stellt eine Herausforderung dar. Es erscheint sinnvoll das Modell auf die eigenen Bedürfnisse/Aufgaben anzupassen. Hierzu gibt es verschiedene Ansätze wie fine-tuning/transfer learning, RAG oder Prompt Engineering. Das Trainieren eines eigenen Modells benötigt viele Daten und ist kostenintensiv aufgrund der benötigten Hardware.

Enorme Hardwareanforderungen beim Trainieren eigener LLMs wie in Table 1 (Patil & Gudivada, 2024, S. 6) von zu sehen. Zur Einordnung, eine H100 Grafikkarte kostet …

**Transfer Learning / Fine-tuning**

~~Laut Géron (2022, S. 373) ist es weniger sinnvoll, ein eigenes tiefes neuronales Netz zu konstruieren, wenn man stattdessen ein bereits vortrainiertes neuronales Netz für eine ähnliche Aufgabe heranzuziehen kann. Dies hat den Vorteil, dass das Training viel schneller erfolgen kann und weniger Daten benötigt werden (Géron, 2022, S. 350). Diese Vorgehensweise nennt sich Transfer Learning (Géron, 2022, S. 350).~~

Einige Organisationen sind der Auffassung, dass das Training eines eigenen Modells nicht rentabel ist. Eine Alternative ist fine-tuning (Lu et al., 2024, S. 2).

Beim Fine-Tuning werden die Gewichte des pre-trained models angepasst anhand eines für eine Aufgabe spezifischen Datensatzes (Brown et al., 2020, S. 6). (Brown et al. (2020, S. 6) führe an, dass solch ein Datensatz dafür typischer Weise tausende bis hunderttausende von beschrifteten Beispiele bedarf. Die Notwendigkeit solcher großen Datensätze mit beschrifteten Daten für jede neue Aufgabe schränkt die Anwendbarkeit von LLMs ein. Für viele Aufgaben ist das Zusammenstellen solcher Datensätze ist kompliziert/aufwändig und müsste für jede Aufgabe wiederholt werden (Brown et al., 2020, S. 3).

Fine-tuning benötigt eine große Menge von den aufgabenspezifischen Daten (Patil & Gudivada, 2024, S. 18).

Fine-Tuning ist günstiger, da ein vortrainiertes Modell verwendet werden kann. Benötigt aber dennoch einige Daten und Rechenkapazitäten.

Im Vergleich zum pretraining ist transfer-learning günstiger und konvergiert schneller, als wenn man das ganze Modell neu trainieren würde (Patil & Gudivada, 2024, S. 18).

~~Pretraining nutzt ungelabelte Daten, folgt dem self-supervised-Ansatz; Transfer-folgt supervised Ansatz mit für die geplante Nutzung gelabelten Daten~~. Pretraining-Datensatz von einer generischen Domäne, wohingegen bei transfer-learning die Daten von einer spezifischen Domäne stammen (Patil & Gudivada, 2024, S. 18).

Für einen ersten Einstieg bleibt somit nur **Prompt-Engineering**.

Ggf. die Inhalte von unten aus „Eingabeaufforderungen“ hier aufführen.

**Vergleichbare Untersuchungen**

In (Maharjan et al., 2024) konnte für Medizinische Benchmarks gezeigt werden, dass mittels Prompt Engineering Techniken ohne fine-tuning SOTA-Ergebnisse für open source Modelle erzielt werden konnte. Dabei wurde deutlich, dass bei fine-tuned Modellen die Zero-Shot Performance besser ist. Yeginbergen et al. (2024, S. 11687) zeigen, dass beim Argument Mining das Finetuning besser abschneidet als Few-shot-Methoden, wobei sich die Art der Aufgabe (Länge und Komplexität der Sequenzspannen) und die Stichprobenmethode als entscheidend erweisen.

Die grundlegenden LLMs eignen sich für allgemeine bzw. übergreifende Tätigkeiten, jedoch weniger für domänenspezifische Aufgaben, wie in den Untersuchungen von Lu et al. (2024) zu entnehmen. Traditionelle Modelle sind für spezifische Aufgaben trainiert (Single-task-learning). LLMs hingegen sind in der Lage mehrere Aufgabe zu übernehmen (multi-task-learning).

Ergebnisse des few-Shot Learnings sind in der Regel schlechter als wenn man ein Modell fine-tunen würden, jedoch werden weniger aufgabenspezifische Daten benötigt (Brown et al., 2020, S. 6).

Argument Mining besteht aus mehreren Teilaufgaben, welche bisher meist (Beispiele aufführen) von einzelnen Modellen übernommen wurden. In dieser Untersuchung soll sich die Fähigkeiten von LLMs zu Nutze gemacht werden, mehrere Aufgaben zu bewältigen.

LLMs benötigen einen riesigen Korpus für das Training. Prompt Engineering benötigt wenig bis gar keine Daten zum trainieren.

**Tokens**

Sanders (2022) beschreibt, dass GPT-Modelle Texte in Form von Tokens verwenden. Demnach entspricht im Englischen ein Token in der Regel einer Länge von einem Zeichen bis zu einem Wort. Die genaue Aufteilung der Texte in Tokens richtet sich nach dem verwendeten Kodierung (engl. encoding) und können von LLM zu LLM abweichen. Ein sogenannter Tokenizer teilt den Text unter Verwendung der Kodierung in eine Liste von Tokens auf. Dies zu verstehen ist relevant für die Arbeit mit LLMs, da einerseits die Modelle nur eine begrenze Anzahl an Token auf einmal verarbeiten können und sich andererseits die Kosten zur Verwendung des Modells nach der übergebenen Tokenanzahl richten.

# Daten und Methoden

- Vorgehen Datengenerierung und -aufbereitung

- Datenlage und -qualität

- Vorgehen und Methoden der Datenanalyse

## Modell / LLM / Large Language Model

Auswahl für Modell(e) begründen. Bspw. ähnliche Kosten als Vergleichskriterium.

Llama 3.2-3B instruct

Abgleich mit Llama 3.2-1B instruct und GPT?

**Gemini 1.5 flash**

Anwendung über LangChain möglich?

**Gpt-4o-mini**

Für die Untersuchung wird das Model GPT-4o mini verwendet. Dieses wird seitens OpenAI (2024b) als ihr kosteneffizientestes kleines Modell ausgewiesen. Nach Angaben von OpenAI (2024b) übertrifft es in akademischen Benchmarks andere LLMs wie Gemini Flash, Claude Haiku und GPT-3.5 Turbo. Das in den gleichen Benchmarks besser abschneidende LLM GPT-4o wurde aufgrund der höheren Kosten und dem begrenzten Budget nicht ausgewählt Quelle für Kosten oder Verweis auf Modellvergleich im Anhang. Eine ausführliche Dokumentation bekräftigt die Entscheidung.

Die Kosten von GPT-4o übersteigen die Kosten von GPT-4o mini zum derzeitigen Stand (01/2025) um … %

Modellbeschreibung

<https://platform.openai.com/docs/models/#gpt-4o-mini>

Das Modell wird über die OpenAI API verwendet.

**Kontextfenster**

Hierbei ist das Kontextfenster des jeweiligen LLMs zu berücksichtigen, welches bei GPT-4o mini bei 128 Tausend Tokens liegt (OpenAI, 2024b). OpenAI (2024c) definieren ein Kontextfenster als einen Wert, welcher die maximale Anzahl an Tokens beschreibt, welche während einer einzigen Anfrage übergeben werden können. Dies beinhaltet sowohl die Input- als auch Output-Tokens sowie Reasoning-Tokens. Input-Tokens sind demnach die Eingabe des Benutzers, Output-Token repräsentieren die vom LLM generierten Antworten und Reasoning-Token werden von dem LLM bei der Generierung einer Antwort genutzt (OpenAI, 2024c).

**Reproduzierbarkeit**

Die Ausgaben eines LLMs können standardmäßig bei gleicher Anfrage unterschiedlich ausfallen (Anadkat (OpenAI), 2023). Um die Ausgaben des LLMs möglichst reproduzierbar werden zu lassen, gibt es seitens OpenAI die Möglichkeit die Modellparameter *seed*, *system\_fingerprint* und beispielsweise *temperature* festzulegen (OpenAI, 2025). Der Parameter *temperature* kann zwischen 0 und 1 festgelegt werden, wobei die Zufälligkeit der Ausgaben des LLMs mit steigendem Wert zunimmt (OpenAI, 2024a). Er wurde für weniger zufällige Ausgaben folglich auf Null festgelegt. Der *system\_fingerprint* ist hingegen eine Kennung des aktuellen Modells inkl. Gewichtungen und weiteren Konfigurationen, wie es von den OpenAI-Servern zur Vervollständigung der Ausgaben genutzt wird (Anadkat (OpenAI), 2023). Diese Kennung und kann sich bei notwendige Änderungen auf der Seite von OpenAI ändern und damit auch die Ausgabe. Bei dem *seed* handelt es sich um eine Ganzzahl, welche, sofern bei den Prompts gleich, in Kombination mit gleichen Modellparametern und gleichem *system\_fingerprint* zu meist identischen Ausgaben des LLMs führt.

Trotz dieser Möglichkeiten wird seitens OpenAI darauf hingewiesen, dass die Konsistenz der Ausgaben verbessert, jedoch nicht garantiert werden kann.

Hier ggf. auf Strukturierte Ausgabe eingehen.

Die restlichen Modellparameter wurden bei den Standardwerten belassen.

## Datensatzbeschreibung

Es gibt verschiedene Datensätze, welche sich in ihrem Schwerpunkt und den Annotationen unterscheiden (Lawrence & Reed, 2020, S. 780–786). Für die Auswahl eines geeigneten Datensatzes wurden diverse Kriterien herangezogen. Zunächst sollte der Datensatz vorab nicht bereits von dem nicht-argumentativen Text befreit worden sein, um die Realität bestmöglich abzubilden (Stab & Gurevych, 2017b, S. 620). Desweiten wird der Ansatz verfolgt, nicht für jede Teilaufgabe des Data Minings einen eigenen Datensatz zu verwenden. Stattdessen soll sich der Datensatz über die drei Teilaufgaben hinweg verwenden lassen. Es bedarf folglich eines annotierten Datensatzes, in dem die Argumentationskomponenten und die argumentativen Beziehungen ausgewiesen werden. Ein bereits annotierter Datensatz ermöglicht es anhand der Grundwahrheit die Ausgaben des LLMs zu evaluieren. Sofern kein passender Datensatz verfügbar ist, bestünde die Möglichkeit einen eigenen Datensatz zu erstellen. Der Schwerpunkt der Untersuchung liegt auf der Anwendung von Eingabeaufforderungen in LLMs für Argument Mining und nicht auf der sprachwissenschaftlichen Theorie zu Argumenten. In Kombination mit der Komplexität der Datenbeschriftung wird sich dagegen die Erstellung eines eigenen Datensatzes entschieden. Zur Orientierung: Die Annotations-Richtlinien von Stab & Gurevych (2017b, S. 630) umfassen 31 Seiten. Entsprechend der zuvor genannten Anforderungen wird der Argument Annotated Essays (Version 2) Datensatz (AAEC) (Stab & Gurevych, 2017a) als geeignet betrachtet. Dieser Datensatz ist das Ergebnis der Arbeit von Stab & Gurevych (2017b) wie sie in dem Artikel „Parsing Argumentation Structures in Persuasive Essays“ beschrieben wird.

Neben dem geringeren zeitlichen Aufwand ist auch die eine höhere Qualität der Annotation von Vorteil.

Der Datensatz besteht aus 420 von der Webseite essayforum.com zufällig ausgewählten überzeugenden Aufsätzen (Stab & Gurevych, 2017b, S. 630). Solche überzeugenden Aufsätze eignen sich gut für das Argument Mining, da sie ein bestimmtes Thema erläutern wobei der Autor versucht die Leser von seinem Standpunkt zu überzeugen (Cabrio & Villata, 2018, S. 5429). Die Annotatoren nutzen zur Beschriftung der Aufsätze das Brat Rapid Annotation Tool. <https://aclanthology.org/E12-2021.pdf> // <http://brat.nlplab.org/>

Zum AAEC

* Annotated with discourse-level argumentation structures.
* Korpus und Annotation-Guidelines frei verfügbar.
* End-to-end argumentation structure parser, that identifies argument components at the token level
* Essays annotated by the expert annotator were used as training data (80%), essays annotated by the other annotators used as test data (20%)

Die Auswahl dieses Datensatzes wird von (Yeginbergen et al., 2024, S. 11688) als “perhaps the most popular NLP dataset manually annotated with argument structures” bezeichnet

Der Datensatz enthält sowohl die vollständigen Aufsätze als Text-Dateien, als auch die Annotationen als ann-Dateien. Bei den Annotationen werden die Argumentationskomponenten Hauptaussage (MajorClaims), Behauptungen (claims) und Prämissen (premises) unterschieden. Nach Stab & Gurevych (2017b, S. 627) beinhalten solche Aufsätze folgende argumentative Struktur. Eine Hauptaussage spiegelt den Standpunkt des Autoren wider, wobei diese anhand von Argumenten unterstützt oder angegriffen werden. Ein Argument besteht aus einer Behauptung und mindestens einer Prämisse. Prämissen sind dabei die Gründe für ein Argument. Um die Haltung der Argumente zu unterscheiden, sind die Behauptungen als für oder gegen markiert. Prämissen hingegen können eine Behauptung oder eine andere Prämisse unterstützen oder angreifen. Es ist möglich, dass es mehrere Hauptaussagen zu einem Text gibt. Hierbei wurde jedoch nicht annotiert, auf welche Hauptaussagen sich die Behauptungen beziehen.

Beschriftung einfügen: Argumentationskomponenten und deren Strukturen. Abbildung ggf. in Anhang packen

Ein Bild, das Text, Diagramm, Reihe, Screenshot enthält.

Automatisch generierte Beschreibung

Eigene Darstellung nach Stab & Gurevych (2017b).

**Datenaufbereitung**

Die IDs geändert, da alle argumentativen Komponenten als TX mit X als fortlaufende Nummer bezeichnet sind. Es wird dabei somit nicht zwischen den Argumentationskomponenten unterschieden. Dieser Schritt wurde vorgenommen, um die Beziehungen besser nachvollziehen zu können, sodass die annotierten Daten dann in JSON überführt werden können. Es wurde sich für ein semi-strukturiertes Format entschieden, da sowohl die Daten, als auch die Ausgabe des LLMs in dieses Format überführt werden können und sie somit für die Evaluation miteinander verglichen werden. Zudem können die Ergebnisse dann für jede Argumentationskomponente individuell betrachtet werden.

## Methode

Im Rahmen der Abschlussarbeit wird ein experimenteller Ansatz verfolgt. Es soll explorativ untersucht werden, inwieweit die Eingabeaufforderungen zur Leistungssteigerung des LLMs für den Anwendungsfall Argument Mining beitragen können. Hierzu werden einem Modell systematisch verschiedene Eingabeaufforderungen übergeben, mit dem Ziel, die Auswirkung der Prompt Engineering Techniken zu untersuchen. Konkret bedeutet dies beispielsweise, dass die Anzahl der Beispiele in einer Eingabeaufforderung variiert wird. Damit dem Modell in der Eingabeaufforderung Beispiele übergeben werden können, werden argumentative Texte benötigt, in denen die Argumentationskomponenten und deren Beziehungen zueinander annotiert sind.

Der zuvor beschriebene Datensatz ist so aufzubereiten, dass daraus die Vorlagen für die Eingabeaufforderungen gemäß der verschiedenen Prompt Engineering Techniken erstellt und an das LLM übergeben werden können. Der Datensatz ist zusätzlich in einen Trainings- und einen Testdatensatz zu unterteilen. Begründung dafür Die Ausgaben des LLMs werden zur Evaluation der Leistung mit der Grundwahrheit abgeglichen. Hierzu sind geeignete Metriken heranzuziehen.

### Tech Stack

Für die Untersuchung werden die Prompts unter Verwendung von Python und speziellen Paketen wie der Transformers Bibliothek von HuggingFace und LangChain systematisch and das LLM übergeben.

### Metriken zur Evaluation

Bei der Wahl einer geeigneten Evaluationsmetrik gibt es aufgrund der vorhandenen Komplexität einige Besonderheiten, die es zu berücksichtigen gilt. So kann es vorkommen, dass das Model eine abweichende Anzahl an Argumentationskomponenten zurück gibt, als tatsächlich vorhanden sind. Das liegt daran, dass es auch nicht-argumentative Textabschnitte gibt oder die Zuordnung der Textabschnitte innerhalb der Argumentationskomponenten vom Modell falsch vorgenommen wurde. Die größere Schwierigkeit besteht darin, dass die vom Modell extrahierten Textabschnitte von der Grundwahrheit (Annotationen) abweichen können, indem mehr oder weniger Wörter der Argumentationskomponente zugeordnet werden. Bei gängigen Klassifikationsmetriken wie Accuracy oder Precision würde ein Textabschnitt, welcher nicht vollständig mit der Grundwahrheit übereinstimmt als eigene Instanz betrachtet werden, zu der keine Grundwahrheit vorhanden ist. Ein Ansatz mit diesem Problem umzugehen ist, Übereinstimmungen oberhalb einer gewissen Grenze als Wahr festzulegen. Ein weitere Ansatz ist die Verwendung von ähnlichkeitsbasierten Metriken.

Metriken, die auf semantischer Ähnlichkeit beruhen, wie BERTScore (Quelle) oder ROGUE (Quelle) werden nicht herangezogen, da die Argumentationskomponenten möglichst exakt und nicht sinngemäß extrahiert werden sollen.

1. Jaccard-Score:
2. BLEU (Bilingual Evaluation Understudy): Metrik, welche die Ausgabe des Modells mit der Grundwahrheit vergleicht, indem die Anzahl der überlappenden Worte ermittelt werden (DataCamp, 2024; Géron, 2022, S. 605). Basiert auf der Präzisionsmetrik (precision) und eignet sich für Textvergleiche.

Der Wert kann zwischen ? und ? liegen. Je höher der Wert, desto höher die Übereinstimmung Quelle. Da die argumentativen Textabschnitte extrahiert und nicht umschrieben werden sollen, erscheint die Metrik als geeignet.

**Konfusionsmatrix**: Berechnung von True Positives (TP), False Positives (FP), False Negatives (FN) und True Negatives (TN), um daraus die Metriken wie Precision und Recall zu bestimmen. Dabei eine Grenze festlegen, ab wann ein Satz nahe genug der Grundwahrheit entspricht und damit als korrekt vorhergesagt gilt.

Hier die Bedeutung der Felder in der Konfusionsmatrix an dem Beispiel von Behauptungen. Diese lassen sich sinngemäß auf die anderen Argumentationskomponenten übertrgen.

* **True Positive (TP):** Die Textabschnitte werden als Behauptung gemäß der Ähnlichkeitsmetrik ausreichend genau erkannt.
* **False Negative (FN)**: Die Textabschnitte werden nicht als Behauptung vom LLM erkannt, obwohl sie es sind.
* **False Positive (FP)**: Die Textabschnitte werden als Behauptung vom LLM identifiziert, obwohl sie es nicht sind.
* **True Negative (TN)**: Die Textabschnitte werden korrekt nicht als Behauptung erkannt. Die Besonderheit bei dem vorliegenden Anwendungsfall ist, dass das LLM nur die argumentativen Texte extrahieren soll. Es gibt folglich keine korrekt vorhergesagten nicht-argumentativen Texte.

Diese Betrachtung erfolgt für jede Argumentationskomponente einzeln. Eine Betrachtung in einer gemeinsamen Konfusionsmatrix wird nicht vorgenommen. Dazu müssten die nicht-argumentativen Texte als solche ebenfalls annotiert werden.

Man kann sich die argumentativen und nicht-argumentativen Textabschnitte in einem Text wie Bausteine vorstellen. Jeder Textbaustein hat feste Grenzen, mit denen er an anliegende Textbausteine angrenzt. Für eine Evaluation im Sinne einer gemeinsamen Konfusionsmatrix müssen für alle Textbausteine mit seinen festen Grenzen eine Klasse vorhergesagt werden und eine Grundwahrheit vorhanden sein. Würde das Modell nun einen Textbaustein nicht korrekt vorhersagen,

Die Aufgabe ist somit nicht nur eine Klassifikation von Textbausteinen, sondern auch die korrekte Textextraktion

Aufgrund des Phänomens von Halluzinationen (Hier noch beschreiben was es ist mit Quelle) bei einem LLM kann es trotz klarer Anweisungen dazu kommen, dass Wörter ergänzt werden oder entfallen. Es würde somit Textbausteine vorliegen die so in dem Text nicht existieren

Aufgrund der zuvor erläuterten möglichen Abweichungen in den Grenzen der extrahierten Textabschnitte würden zusätzliche Komplikationen entstehen.

Bei eine sauberen Klassifizierung der Textabschnitte eines Aufsatzes in die drei Argumentationskomponenten und nicht-argumentativen Textabschnitte ist jedes Wort einer der Klassen zuzuordnen.

Der damit verbundene Aufwand wird als zu hoch bewerten, mit keiner zu geringen Erfolgsaussicht.

Nichtsdestotrotz lassen sich aus den Werten der Konfusionsmatrix diejenigen Klassifikationsmetriken berechnen, für die FN nicht benötigt werden. Somit können Precision, Recall und F1-Score berechnet werden. Im Hinblick auf vergleichbare Untersuchungen ist der F1-Score die

“Mochales Palau and Moens perform machine learning, using a variety of features on different levels of description. Their implementation proceeds in two steps: First, sentences are being classified as either ‘argumentative’ or ‘non-argumentative’. In their implementation with a multinomial naive Bayes classifier and a maximum-entropy model, their best average accuracy was almost 74%. […] The second step tries to further classify the ‘argumentative’ sentences into the categories of ‘premise’ and ‘conclusion’, and here they achieved F-measures of 68% and 74%, respectively.” (Peldszus & Stede, 2013, S. 25–26)

Mochales Palau, R., & Moens, M.-F. (2009). Argumentation mining: The detection, classification and structure of arguments in text. In Proceedings of the ICAIL 2009, Barcelona, Spain (pp. 98–109)

* F1 macro-averaged score calculated at sequence level (Yeginbergen et al., 2024, S. 11691) “Precision is the percentage of named entities found by the learning system that are correct. Recall is the percentage of named entities present in the corpus that are found by the system. A named entity is correct only if it is an exact match of the corresponding entity in the data file.” (Tjong Kim Sang and De Meulder, 2003).
* Stance Classification 🡪 Macro F1
* Stab & Gurevych (dataset) 🡪 macro F1-Score für Components (MC, C, Pr), Relations (Linked, not linked) und Stances (Support, Attack)

*„Es ist wichtig, hier zu betonen, dass auch die menschliche Zustimmung (die im Allgemeinen als Obergrenze für die automatische Leistung bei Annotationsaufgaben angesehen wird) von der Komplexität der AM-Aufgaben beeinflusst wird“* (Cabrio & Villata, 2018, S. 5431).

### Eingabeaufforderungen / Prompts

Anstatt Modelle für jede einzelne Teilaufgabe zu trainieren, soll sich die Lernfähigkeit von LLMs zunutze gemacht werden.

LLMs benötigen jedoch in der Regel große Datensätze, um ein Modell mit ausreichender Performance zu trainieren. Nicht immer besteht jedoch die Möglichkeit, einen großen Datensatz mit annotierten Daten zusammenzustellen, da dies zeitaufwändig und kostspielig sein kann (Tunstall et al., 2023, S. 289), wie auch in dem Fall von Argument Mining (Lawrence & Reed, 2020, S. 780).

Während des Vortrainings entwickelt ein Sprachmodell Mustererkennungsfähigkeiten (Brown et al., 2020, S. 3). Brown et al. (2020, S. 3–6) sowie Wei et al. (2023) zeigen, dass diese Fähigkeiten dann bei Eingabeaufforderungen genutzt werden können, um das Modell anhand von Beispielen an die gewünschte Aufgabe anzupassen. Dies wird als In-Context Learning (ICL) bezeichnet. Die Gewichtungen des Modells werden dabei nicht verändert. ICL kann in Zero-Shot Learning, One-Shot Learning und Few-Shot Learning unterschieden werden (Patil & Gudivada, 2024, S. 23–25; Tunstall et al., 2023, S. 189). Die Unterscheidung richtet sich danach, wie viele Beispiele in der Eingabeaufforderung übergeben werden.

Prompt Engineering baut darauf auf, dass LLMs in der Lage sind Aufgaben zu erfüllen, für welche sie nicht explizit trainiert wurden und beschäftigt sich mit der Gestaltung von Eingabeaufforderungen, welche genutzt werden, um die Fähigkeiten von LLMs für bestimmte Aufgaben zu nutzen (Trad & Chehab, 2024, S. 369). Maharjan et al. (2024) konnten für medizinische Benchmarks zeigen, dass mittels Prompt Engineering Techniken ohne Fine-Tuning dem Stand der Technik entsprechende Ergebnisse für Open-Source Modelle erzielt werden können. Dem gegenüber stehen Untersuchungen wie Trad und Chehab (2024) oder (Yeginbergen et al., 2024), welche zu dem Ergebnis kommen, dass Prompt Engineering eine schlechtere Leistung hervorbringt als Fine-Tuning.

Nichtsdestotrotz bleibt es aufgrund seiner Einfachheit und Flexibilität eine attraktive Alternative, insbesondere bei Anwendungsfällen, in denen die hohen Anforderungen von LLMs an Datenmenge und Hardware die Anwendung von Fine-Tuning unmöglich machen. Deshalb soll das Potential von Prompt Engineering speziell für das Argument Mining untersucht werden, um zu evaluieren, ob es trotz der beschriebenen Einschränkungen eine praktikable und ressourcenschonende Lösung für diese Aufgabe bieten kann.

**Zero-Shot Learning/Prompting (ZS):** Beim ZS werden in dem Prompt kein Beispiel aufgeführt. Dem Modell wird lediglich eine Beschreibung der Aufgabe in natürlicher Sprache übergeben (Brown et al., 2020, S. 7).

**One-Shot Learning (OS):** Beim OS wird in dem Prompt hingegen neben der Aufgabenbeschreibung zusätzlich ein Beispiel aufgeführt (Brown et al., 2020, S. 6). Das Beispiel wird als Kombination von übergebenen Input und gewünschtem Output aufgestellt. Der Input ist diesem Fall der Text des Aufsatzes und der Output die Argumentationskomponenten sowie deren Beziehungen, strukturiert als JSON Objekt. Die Beispiele können zufällig oder systematisch ausgewählt werden. Hier die gewählte Vorgehensweise beschreiben und begründen

Das Beispiel wird zufällig aus dem Datensatz ausgewählt vs. Aufgrund der Tokenanzahl wird das Beispiel mit der geringsten Tokenanzahl ausgewählt.

**Few-shot Learning/Promping (FS):** Auch hier werden wie beim OS dem Modell zusätzlich zur Aufgabenbeschreibung Beispiele als Input-Output Paare übergeben. Nach Brown et al. (2020, S. 6, 10) werden dabei in der Regel zwischen 10 und 100 Beispiele übergeben, je nach der Größe des Kontextfensters des LLMs. Demnach führen mehr Beispiele meist, aber nicht immer, zu besseren Ergebnisse. So weisen Google (2024) darauf hin, dass Experimente notwendig sind um die optimale Anzahl der Beispiel zu bestimmen. Denn die Übergabe von zu vielen Beispielen kann auch zum Overfitting führen.

Few-Shot-Learning ermöglicht es dem LLM sich mit wenigen Trainingsdaten auf die neue Aufgabe anzupassen (Ozdemir, 2024, S. 136).

Das Hauptziel von Few-Shot Learning (FSL) ist die Verallgemeinerung beim Lernen aus einem kleinen Teil der Daten (Yeginbergen et al., 2024, S. 11690).

Yeginbergen et al. (2024, S. 11690) weisen darauf hin, dass bei dem Argument Mining auch Beispiele aufgenommen werden sollen, welche keine Argumentationskomponenten beinhalten. Da der Datensatz lediglich argumentative Aufsätze beinhaltet, werden keine nicht-argumentativen Texte als Beispiele übergeben. Die Aufsätze enthalten allerdings auch nicht-argumentative Textstellen, welche keine Argumentationskomponenten darstellen.

Große Texte mit vielen Tokens erfordern mehr Rechenressourcen, längere Verarbeitungszeiten und Kosten somit auf mehr. Deshalb ist ein effizienter Umgang mit der Tokenanzahl ein wichtiger Aspekt bei der Anwendung von LLMs.

Das Problem ist, das die maximalen Tokenanzahl für die HuggingFace API bei 4096 liegt. Das bedeutet, dass in Summe die Eingabe Token und die Ausgabe Tokens diesen Wert nicht überschreiten dürfen. Eingabeaufforderungen oberhalb dieser Begrenzung landen in einer Fehlermeldung. Die maximale Ausgabetokenanzahl muss somit auch so gewählt werden, sodass die Ausgaben nicht zu früh abgebrochen werden. Dies hat ebenfalls zur Folge, dass in den Eingabeaufforderungen nicht unbegrenzt viele Beispiele übergeben werden können. Bei beispielsweise 1000 Ausgabetoken dürfen nur noch 3000 Token eingegeben werden. Dies wird bereits bei 3 Beispielen überschritten. Es ist somit notwendig eine andere Möglichkeit zu finden, um dem Daten die relevanten Informationen zu übergeben, ohne diese Grenze zu überschreiten. Dafür wird das Kontext-Fenster des Modells herangezogen, welches mit 128 Tsd. Token Quelle größer ist, als die Grenze bei Eingabeaufforderungen.

Das Modell besitzt jedoch ein größeres Kontext-Fenster Quelle. Ggf. kann das ausgenutzt werden, indem die Beispiele nacheinander übergeben werden.

ODER Teilaufgaben in aufeinanderfolgenden Prompts ausführen lassen. Der Output eines Prompts wird von dem darauffolgenden Prompt genutzt (refinement, prompt chaining, sequential few-shot prompting, iterative re-prompting).

Ansätze wie die automatische Begrenzung der Eingabeaufforderung (quelle) bei erreichen der Token-Grenze würden wesentliche Informationen beschneiden. Auch die Unterteilung der Essays in Teilabschnitte würde das gesamtheitliche Verständnis des Textes verzerren.

Neben der Ergänzung von Beispielen in den Prompts gibt es noch weitere Ansätze. Dazu gehören beispielsweise Chain-of-thought prompting und Role Playing.

* **Chain-of-thought prompting (COT):** Wei et al. (2023, S. 2) definieren COT als eine Reihe von Zwischenschritten in natürlicher Sprache, die zu dem Ergebnis führen. Wei et al. (2023) zeigen, wie COT die Leistung des Modells bei komplexen Logikaufgaben signifikant ohne fine-tuning verbessern kann. Dazu sind dem Modell Beispiele für COT in dem Prompt zu übergeben, wie beim One-Shot bzw. Few-Shot Learning. Auf den Anwendungsfall Argument Mining übersetzt werden dem Modell die Teilaufgaben genannt und die Vorgehensweise anhand eines Beispiels dargestellt.
* **Role Playing / Persona:** Hierbei wird das LLM angehalten eine gewisse Persona zu imitieren und die Ausgaben entsprechend zu formulieren, um so relevante Informationen auszugeben (OpenAI, 2024e; Trad & Chehab, 2024, S. 369).

Neben den bereits genannten Techniken empfehlen Google (2024) und OpenAI (2024e) für bessere Ergebnisse beispielsweise das Schreiben von spezifischen Anweisungen mit Kontextinformationen, die konsistente Formatierung von Beispielen und die Verwendung von Begrenzungszeichen sowie den systematischen Test von Veränderungen in den Prompts. Dies stellt nur eine Auswahl möglicher Techniken dar. Einzelne Ansätze wie Self-Consistency, bei welchem zu einem Prompt mehrere Ausgaben erzeugt und die am häufigsten vorkommende Antwort verwendet wird (Meta, 2024), werden für die Untersuchung nicht betrachtet.

Die Prompts werden modular anhand von Textbausteinen erstellt. Sie enthalten die Aufgabenbeschreibung, das Ausgabeformat, die schrittweise Aufgabenbeschreibung und die Beschreibung der Persona. Diese Textbausteine können dann um Beispiele ergänzt und miteinander kombiniert werden. Damit soll sichergestellt werden, dass leichte Abweichungen in der Formulierung die Ergebnisse nicht verzerren. Darüber hinaus lassen sich so flexibel weitere Prompts konstruieren.

Dem Modells wird JSON als Ausgabeformat auferlegt. Dadurch soll sich das Modell bei der Ausgabe auf die wesentlichen Informationen beschränken (OpenAI, 2024f). Das Ausgabeformat wurde so definiert, dass es sich für die Weiterverarbeitung zur anschließenden Evaluation eignet. Die Eingabeaufforderungen sind in Englisch formuliert, da der Datensatz englische Texte beinhaltet. Die multilingualen Fähigkeiten des LLMs sind nicht Teil dieser Untersuchung.

Konkret werden dem LLM im Sinne des ZS, OS und FS eine Aufgabenbeschreibung mit keinem, einem oder mehreren Beispielen übergeben. Diese grundlegende Prompt-Struktur wird um den Textbaustein Persona oder COT oder beiden ergänzt. Hieraus ergeben sich 12 Prompts, anhand derer die Auswirkungen der übergebenen Beispiele und der Ansätze Verwendung einer Persona sowie COT analysiert werden können. Da X Texte als Beispiele verwendet werden und somit als Trainingsdaten zählen, können zur Evaluation … Texte herangezogen werden.

# Ergebnisse

- Möglichst objektive Darstellung der Ergebnisse

- Idealerweise 1-3 zentrale Visualisierungen der Kernaussagen. Pro-Tipp: Aussagestarke Grafik mit ausführlicher Tabelle im Anhang

Plot von Evaluationsmetrik in Abhängigkeit des Token Counts (=Kosten) pro Prompt Technik. Es wurde sich explizit für die Darstellung in Abhängigkeit der Tokenanzahl anstelle der tatsächlichen kosten entschieden, da die Kosten pro Token individuell für jedes Modell sind. Die Tokenanzahl kann als Kostentreiber herangezogen werden.

Bspw. Kennzahl bilden: Genauigkeit pro Token?

# Diskussion und Handlungsempfehlungen

- Diskussion der Ergebnisse

- Möglicherweise Ableitung von Handlungsempfehlungen

- Limitationen, weiterführende (neue) Forschungsfragen, etc.

Entlang der Untersuchung wurde deutlich, dass es sich beim Argument Mining um ein komplexes Thema handelt, …

**Praktischer Mehrwert**

Die geplante Abschlussarbeit wird voraussichtlich Aufschluss darüber geben, wie leistungsfähig LLMs für das Argument Mining sind und welche Prompt Engineering Techniken in diesem Zusammenhang die besten Ergebnisse liefern. Sie würde sowohl zur Weiterentwicklung der Forschung auf dem Gebiet des Argument Minings als auch zur praktischen Anwendung von LLMs in realen Anwendungsfällen beitragen. Eine effiziente und strukturierte Extraktion von Argumenten kann dazu beitragen, Diskussionen nachvollziehbarer zu machen, sodass komplexe Themen besser durchdrungen und darauf aufbauend fundierte Entscheidungen getroffen werden können.

**Limitationen**

* Die Ergebnisse gelten nur für die untersuchten LLMs
* Nur für englische Sprache
* Sowohl die Verwendung von abweichenden Formulierungen, als die Reihenfolge des Inhalts kann zu unterschiedlichen Antworten des LLMs führen (Google, 2024). Die Ergebnisse beziehen sich folglich auf die dargestellten Prompt Templates.
* Der Datensatz enthält lediglich argumentative Aufsätze, weshalb die Prompts auch nur Beispiele von argumentativen Aufsätzen beinhalten. Für andere argumentative Textarten oder auch nicht-argumentative Texte sind weitere Untersuchungen durchzuführen.

**Handlungsempfehlungen**

* Untersuchung für weitere LLMs
* Untersuchung der Performance, wenn theoretische Wissen wie die Annotation Richtlinien mit RAG eingebaut werden.
* Anwendung von Fine-Tuning.
* Untersuchung für weitere Sprachen
* Weiterentwicklung indem Argumentationsstrukturen in Strukturdiagrammen aufgeführt werden, um die Extrahierten Teile visuell leicht verständlich aufzuarbeiten.
* Für ein Tiefgreifendes Verständnis der Fehler des Models (Error Analysis) wäre eine gemeinsame Confusion Matrix für Major Claims, Claims und Premises hilfreich. Somit könnte man sehen bei welchen Argumentationskomponenten es eher zu verwechslungen kommt. Dazu müssten jedoch die NCHT-Argumentativen Textabschnitte als solche annotiert werden. Hier begründen, warum es nicht gemacht wurde.

# Fragestellung

Wie können LLMs auf eine spezifische Aufgabe traininiert werden

1. Wie gut sind LLMs für argument Mining geeignet (Identifizierung von claims, premises und conclusions) bei der Anwendung von Zero-shot und few-shot learning
2. Können LLMs argumentative Strukturen (claim-premise-pairs) zuverlässig erkennen ?
3. Welche Herausforderungen gibt es bei der Anwendung von LLMs fürs Argument Mining? (Verschiedene Bestandteile von Argumenten, Mehrere Teilaufgaben,…)
4. Welche Prompt Engineering Techniken sind am Effektivsten zur Verbesserung der Perfromance eines LLMs zum Argument Mining?
5. Wie beeinflusst die Anzahl der übergebenen Beispiele die Leistung des LLMs?
6. Wie unterscheiden sich die Modelle mit unterschiedlichen Strukturen) dabei untereinander? Gemini, GPT4o, GPT2?
7. Wie effektiv sind LLMs bei der Anpassung an verschiedene Argumentationssstile bei verschiedenen Domänen (Politik, Recht, Wissenschaft)

# Literaturverzeichnis / Quellenverzeichnis

Anadkat (OpenAI), S. (2023). *How to make your completions outputs consistent with the new seed parameter*. https://cookbook.openai.com/examples/reproducible\_outputs\_with\_the\_seed\_parameter

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). *Language Models are Few-Shot Learners* (arXiv:2005.14165). arXiv. http://arxiv.org/abs/2005.14165

Cabrio, E., & Villata, S. (2018). Five Years of Argument Mining: A Data-driven Analysis. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 5427–5433. https://doi.org/10.24963/ijcai.2018/766

Cheng, L., Bing, L., He, R., Yu, Q., Zhang, Y., & Si, L. (2022). IAM: A Comprehensive and Large-Scale Dataset for Integrated Argument Mining Tasks. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2277–2287. https://doi.org/10.18653/v1/2022.acl-long.162

DataCamp. (2024). *LLM Evaluation: Metrics, Methodologies, Best Practices*. DataCamp. https://www.datacamp.com/blog/llm-evaluation

Géron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (3. Aufl.). O’Reilly Media, Inc.

Google. (2024). *Prompt design strategies*. https://ai.google.dev/gemini-api/docs/prompting-strategies

Kochmar, E. (2022). *Getting started with Natural Language Processing*. Manning Publications.

Lawrence, J., & Reed, C. (2020). Argument Mining: A Survey. *Computational Linguistics*, *45*(4), 765–818. https://doi.org/10.1162/coli\_a\_00364

Lu, R.-S., Lin, C.-C., & Tsao, H.-Y. (2024). Empowering Large Language Models to Leverage Domain-Specific Knowledge in E-Learning. *Applied Sciences*, *14*(12), 5264. https://doi.org/10.3390/app14125264

Maharjan, J., Garikipati, A., Singh, N. P., Cyrus, L., Sharma, M., Ciobanu, M., Barnes, G., Thapa, R., Mao, Q., & Das, R. (2024). OpenMedLM: Prompt engineering can out-perform fine-tuning in medical question-answering with open-source large language models. *Scientific Reports*, *14*(1), 14156. https://doi.org/10.1038/s41598-024-64827-6

Meta. (2024). *Prompting*. How-to Guides. https://www.llama.com/docs/how-to-guides/prompting/

OpenAI. (2024a). *Chat. Temperature*. OpenAI Platform. API Reference. https://platform.openai.com/docs/api-reference/chat/create

OpenAI. (2024b). *GPT-4o mini: Advancing cost-efficient intelligence*. Openai.Com. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

OpenAI. (2024c). *Models*. OpenAI Platform. https://platform.openai.com/docs/models/

OpenAI. (2024d). *Optimizing LLM Accuracy*. OpenAI Platform. Docs. https://platform.openai.com/docs/guides/optimizing-llm-accuracy

OpenAI. (2024e). *Prompt engineering*. OpenAI Platform. Docs. https://platform.openai.com/docs/guides/prompt-engineering

OpenAI. (2024f). *Structured Outputs*. Docs. https://platform.openai.com/docs/guides/structured-outputs

OpenAI. (2025). *Advanced usage*. OpenAI Platform. https://platform.openai.com

Ozdemir, S. (2024). *Praxiseinstieg Large Language Models: Strategien und Best Practices für den Einsatz von ChatGPT und anderen LLMs* (F. Langenau, Übers.; 1. Aufl., deutsche Ausgabe). O’Reilly.

Patil, R., & Gudivada, V. (2024). A Review of Current Trends, Techniques, and Challenges in Large Language Models (LLMs). *Applied Sciences*, *14*(5), 2074. https://doi.org/10.3390/app14052074

Peldszus, A., & Stede, M. (2013). From Argument Diagrams to Argumentation Mining in Texts: A Survey. *International Journal of Cognitive Informatics and Natural Intelligence*, *7*(1), 1–31. https://doi.org/10.4018/jcini.2013010101

Sanders, T. (2022, Dezember 16). *How to count tokens with Tiktoken*. OpenAI Cookbook. https://cookbook.openai.com/examples/how\_to\_count\_tokens\_with\_tiktoken

Stab, C., & Gurevych, I. (2014). *Annotating Argument Components and Relations in Persuasive Essays*.

Stab, C., & Gurevych, I. (2017a). *Argument Annotated Essays (version 2)* [Dataset]. https://tudatalib.ulb.tu-darmstadt.de/handle/tudatalib/2422

Stab, C., & Gurevych, I. (2017b). Parsing Argumentation Structures in Persuasive Essays. *Computational Linguistics*, *43*(3), 619–659. https://doi.org/10.1162/COLI\_a\_00295

Trad, F., & Chehab, A. (2024). Prompt Engineering or Fine-Tuning? A Case Study on Phishing Detection with Large Language Models. *Machine Learning and Knowledge Extraction*, *6*(1), 367–384. https://doi.org/10.3390/make6010018

Tunstall, L., Werra, L. von, Wolf, T., & Géron, A. (2023). *Natural Language Processing mit Transformern: Sprachanwendungen mit Hugging Face erstellen* (M. Fraaß, Übers.; 2. Aufl.). O’Reilly.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2023). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models* (arXiv:2201.11903). arXiv. http://arxiv.org/abs/2201.11903

Yeginbergen, A., Oronoz, M., & Agerri, R. (2024). Argument Mining in Data Scarce Settings: Cross-lingual Transfer and Few-shot Techniques. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 11687–11699. https://doi.org/10.18653/v1/2024.acl-long.628

# Anhänge & Projektdokumentation

Projektdokumentation umfasst:

* Readme-Datei
* Instrumente
* Daten
* EDA
* Qualitätssicherung?
* Code der Datenaufbereitung
* Coder der Datenanalyse
* Weiterführende Materialien

## Modellvergleich

Modellvergleich als Tabelle aufarbeiten und in Anhang packen

|  |  |  |  |
| --- | --- | --- | --- |
| Modell | Beschreibung | API-Kosten in USD pro 1M Token | Link |
| Gemini 1.5 Flash | Our fastest multimodeal model with great perfromance for diverse, repetative tasks and a 1 million context window. | Prompts up to 128 tokens  Input: 0,075 $  Output: 0,3 $  Context Caching: 0,01875 $  Prompts longer than 128 tokens  Input: 0,15 $  Output: 0,6 $  Context Caching: 0,0375 $ | https://ai.google.dev/pricing#1\_5flash |
| Gemini 1.5 Flash-8B | Our smales model for lower intelligence use cases with a 1 million token context window | Prompts up to 128 tokens  Input: 0,0375 $  Output: 0,15 $  Context Caching: 0,001 $  Prompts longer than 128 tokens  Input: 0,075 $  Output: 0,3 $  Context Caching: 0,02 $ | https://ai.google.dev/pricing#1\_5flash-8B |
| Gemini 1.5 Pro | Our next generation model with a breaktrhough 3 million context window | Prompts up to 128 tokens  Input: 1,25 $  Output: 5 $  Context Caching: 0,3125 $  Prompts longer than 128 tokens  Input: 2,5 $  Output: 10,0 $  Context Caching: 0,625 $ | https://ai.google.dev/pricing#1\_5pro |
| Claude 3.5 Sonnet | Out most intelligent model to date  200k Context window | Prompt write  3,75 $  Prompt read  0,3 $ | https://www.anthropic.com/pricing#anthropic-api |
| GPT-4o | GPT-4o is our most advanced multimodal model that’s faster and cheaper than GPT-4 Turbo with stronger vision capabilities. The model has 128K context and an October 2023 knowledge cutoff. | Input: 2,5 $  Output: 10 $  Cached Input: 1,25 $  With Batch API  Input: 1,25 $  Output: 5 $ | https://openai.com/api/pricing/ |
| GPT-4o mini | GPT-4o mini is our most cost-efficient small model that’s smarter and cheaper than GPT-3.5 Turbo, and has vision capabilities. | Input: 0,15 $  Output: 0,6 $  Cached Input: 0,075 $  With Batch API  Input: 0,075 $  Output: 0,3 $ | https://openai.com/api/pricing/ |
| OpenAI o1 | o1 is our most powerful reasoning model that supports tools, Structured Outputs, and vision. The model has 200K context and an October 2023 knowledge cutoff. | Input: 15 $  Output: 60 $  Cached Input: 7,5 $  With Batch API  entfällt | https://openai.com/api/pricing/ |
| OpenAI o1 mini | o1-mini is our small reasoning model that thinks faster than o1 and is optimized for coding and math. | Input: 3 $  Output: 12 $  Cached Input: 1,5 $  With Batch API  entfällt | https://openai.com/api/pricing/ |
| Llama small 0B-8B |  | $0.4 | https://www.llama-api.com/pricing |
| Llama small 8B-30B |  | $1.6 | https://www.llama-api.com/pricing |
| Llama small >30B |  | $2.8 | https://www.llama-api.com/pricing |

## Prompt Templates

Hier die Prompts abbilden. Zum einen schematisch aus dem beschreibenen Textbausteinen und zum anderen die TXT-Dateien.