Digital Business University of Applied Sciences

Masterarbeit zum Thema:

**Der Einfluss von Prompt Engineering auf Large Language Models**

**Im Argument Mining**

zur Erlangung des Grades Master of Science

|  |  |
| --- | --- |
| **Eingereicht von** |  |
| Vorname Name | Benjamin Fels |
| Email | benjamin.fels@student.dbuas.de |
|  |  |
| Matrikelnummer: | 190200 |
| Fachsemester: | 5 |
| Studiengang: | Data Science & Management |
|  |  |
| Datum Abgabe: | 31.01.2025 |
|  |  |
| Gutachter: | Prof. Dr. Marcel Hebing |

Executive Summary

Eine vollständige Zusammenfassung mit Fokus aus actionable insights

Inhaltsverzeichnis

[1 Einleitung 1](#_Toc188683511)

[2 Daten und Methoden 6](#_Toc188683512)

[2.1 Large Language Model 6](#_Toc188683513)

[2.2 Datensatz 8](#_Toc188683514)

[2.3 Methode 11](#_Toc188683515)

[2.3.1 Prompts 12](#_Toc188683516)

[2.3.2 Metriken zur Evaluation 14](#_Toc188683517)

[3 Ergebnisse 18](#_Toc188683532)

[4 Diskussion und Handlungsempfehlungen 20](#_Toc188683533)

[5 Fragestellung 24](#_Toc188683534)

[6 Literaturverzeichnis / Quellenverzeichnis 25](#_Toc188683535)

[7 Anhänge & Projektdokumentation 30](#_Toc188683536)

[7.1 Modellvergleich 30](#_Toc188683537)

[7.2 Prompt Templates 32](#_Toc188683538)

Abbildungsverzeichnis

[Abbildung 1 Beispiel für argumentative Struktur der Aufsätze 10](#_Toc188790283)

[Abbildung 2 Durchschnittlicher F1-Score für Argumentationskomponenten und Beziehungen pro Prompt 18](#_Toc188790284)

[Abbildung 3 Zero-Shot Prompt-Struktur 32](#_Toc188790285)

[Abbildung 4 One-Shot Prompt-Struktur 32](#_Toc188790286)

Abkürzungsverzeichnis

|  |  |
| --- | --- |
| **Abkürzung** | **Bedeutung** |
| OS | One-Shot |
| ZS | Zero-Shot |
| FS | Few-Shot |
| FS-x (z.B. FS-10) | Few-Shot mit x-Beispielen |
| COT | Chain-of-thought |
| LLM | Large Language Model |
| NLP | Natural Language Processing |
| ICL | In-Context-Learning |
| TP | True Positive |
| FN | False Negative |
| FP | False Positive |
| TN | True Negative |

# Einleitung

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

- Kontext der Arbeit klar verständlich machen

- Entwicklung der Forschungsfrage(n)

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). 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. 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 diesem Prozess beschäftigt, nennt sich Argument Mining.

**Argument Mining**

Unter Argument Mining kann 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

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; Peldszus & Stede, 2013, S. 20; 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 zwischen den Argumentationskomponenten identifiziert. Da die Teilaufgaben aufeinander aufbauen, wirken sich Fehler am Anfang negativ auf die nachfolgenden Aufgaben aus (Stab & Gurevych, 2017b, S. 648–649). Klassische Ansätze für Argument Mining setzen häufig auf umfangreiche regelbasierte Verfahren oder spezialisierte maschinelle Lernmodelle, die auf spezifische Datensätze trainiert werden (Lawrence & Reed, 2020; Stab & Gurevych, 2017b). Nach Cabrio & Villata (2018, S. 5431) werden dabei syntaktische und positionsbezogene Merkmale am häufigsten verwendet. Neuere Argument Mining Ansätze betrachten die Extraktion der Argumente als eine Sequenzetikettierungsaufgabe (engl. sequence labeling task), vergleichbar mit der Named Entity Recognition (Cheng et al., 2022, S. 2282; Stab & Gurevych, 2017b, S. 636; Yeginbergen et al., 2024, S. 11688). Ein Modell für jede Teilaufgabe des Argument Minings zu entwickeln ist mit einem hohen Aufwand und Fachwissen verbunden, wie es beispielsweise aus Stab & Gurevych (2017b) hervorgeht.

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 Informationsextraktion (Han et al., 2024, S. 5; Kochmar, 2022). Sprachmodelle können als Modelle verstanden werden, welche die Abfolge von Token vorhersagen (Patil & Gudivada, 2024, S. 4). Dabei können Token einzelne Buchstaben bis hin zu ganzen Wörtern umfassen (Sanders, 2022). Nach Han et al. (2024, S. 11) bezieht sich die Bezeichnung großes Sprachmodell auf tiefe neuronale Netze mit mehr als einer Milliarde Parametern. Demnach besitzen sie starke Generalisierungsfähigkeiten, die es ihnen ermöglichen auf ein breites Spektrum an Aufgaben angewendet zu werden (Han et al., 2024, S. 34). Bekannte LLMs sind die GPT-Reihe von OpenAI oder die LLaMA-Reihe von Meta (Han et al., 2024, S. 27). Patil & Gudivada (2024, S. 3) unterscheiden drei Phasen bei LLMs: Pre-training, Transfer-Learning und In-Context-Learning*.* Nach Han et al. (2024, S. 42) wird beim Pre-training das Model auf einem vielfältigen Datensatz trainiert, damit es eine gute Generalisierungsfähigkeit entwickelt. Während dieser Phase entwickelt ein LLM Mustererkennungsfähigkeiten (Brown et al., 2020, S. 3). Transfer Learning beschreibt hingegen die Anwendung des LLMs auf einen neuen Anwendungsfall (Géron, 2022, S. 6, 350; Patil & Gudivada, 2024, S. 3). Eine spezielle Form des Transfer-Learning ist das Fine-Tuning, wobei mithilfe von aufgabenspezifischen Daten die ursprünglichen Parameter des vortrainierten Modells aktualisiert werden (Brown et al., 2020, S. 6; Han et al., 2024, S. 24; Patil & Gudivada, 2024, S. 18). Bei der dritten Phase In-Context Learning (ICL) wird sich die Generalisierungsfähigkeit eines LLMs zunutze gemacht. Brown et al. (2020, S. 3–6) sowie Wei et al. (2023) zeigen, das mittels sogenannter Prompts ein LLM anhand von Beispielen an die gewünschte Aufgabe angepasst werden kann. Ein Prompt kann als ein Eingabetext verstanden werden, auf welcher das LLM reagiert. Solch ein Eingabetext kann beispielsweise Fragen oder Anweisungen enthalten. Der Begriff ICL beschreibt die Anpassungsfähigkeit des LLMs anhand solch eines Prompts (Brown et al., 2020, S. 3). Die Parameter des Modells werden dabei nicht verändert. Hierfür werden wesentlich weniger aufgabenspezifische Daten benötigt (Brown et al., 2020, S. 6). Konkret werden nur so viele Daten gebraucht, wie Beispiele in den Prompts aufgeführt werden. Die Anzahl der Beispiele kann nach Brown et al. (2020, S. 10) zwischen Null bis zu dem maximal zulässigen Wert des Kontextfensters des LLMs gewählt werden, was typischer Weise zwischen 10 und 100 Beispielen liegt. Prompt Engineering baut auf dieser Fähigkeit des ICL auf und umfasst den Prozess der Gestaltung von Prompts, um die gewünschten Ausgaben von dem LLM zu erhalten (Patil & Gudivada, 2024, S. 20; Trad & Chehab, 2024, S. 369).

Nach Patil & Gudivada (2024, S. 31) benötigen solche LLMs für das Pre-training tausende an GPUs für mehrere Wochen. Neben den Kosten für die Hardware kommen die Kosten für die benötigte Energy, das Fachpersonal und die Infrastruktur zur Verwendung des LLMs hinzu. Dieser Ansatz ist folglich unbezahlbar für eine Vielzahl von Forschenden (Patil & Gudivada, 2024, S. 31). Die Anpassung eines bereits vortrainierten LLMs mittels Fine-Tuning für die eigene Anwendung ist hingegen nach Patil & Gudivada (2024, S. 18) kostengünstiger. Brown et al. (2020, S. 6) führen jedoch an, dass für das Fine-Tuning ein Datensatz in mit einem Umfang von typischer Weise tausende bis hunderttausende von beschrifteten Beispielen bedarf. Die Notwendigkeit solch großer Datensätze schränkt die Anwendbarkeit von LLMs ein, da nicht für jede spezifische Aufgabe ausreichend annotierte Daten in der erforderlichen Menge verfügbar sind und eine Beschaffung zeitaufwendig und kostenintensiv sein kann (Tunstall et al., 2023, S. 289). Zudem müsste für jede Aufgabe erneut ein LLM mittels Fine-Tuning angepasst werden, wofür wiederum jeweils eigene ausreichend große Datensätze benötigt werden würden (Brown et al., 2020, S. 6; Patil & Gudivada, 2024, S. 18). Daraus folgt, dass auch wenn dieser Ansatz kostengünstiger ist, er sich nicht für Fälle eignet, in denen keine ausreichenden Datensätze vorhanden sind. Nach den Beschreibungen von Lawrence & Reed (2020, S. 780, 798) liegt solch ein Mangel an entsprechend annotierten Daten im Argument Mining vor und stellt eine große Herausforderung für das Forschungsgebiet dar.

Der Prompt-basierte Ansatz bietet eine interessante Alternative, da hierbei weniger Daten benötigt werden und ein einziges LLM für mehrere Anwendungsfälle verwendet werden kann (Brown et al., 2020, S. 6; Patil & Gudivada, 2024, S. 20). Untersuchungen, in denen die Auswirkung von Fine-Tuning und Prompt Engineering auf die Leistung eines LLMs untersucht wurden, kommen zu unterschiedlichen Ergebnissen. 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. Auch Brown et al. (2020, S. 6) führen an, dass die Ergebnisse bei der Verwendung von ICL in der Regel schlechter sind, als die eines mittels Fine-Tuning angepassten LLMs, jedoch mit dem Hinweis, dass weniger aufgabenspezifische Daten benötigt werden. OpenAI (2024d) betont, dass Prompt Engineering bei der Anwendungen von LLMs auf eigene Anwendungsfälle als ersten Ansatz gewählt werden sollte. Erst sofern die Ergebnisse nicht ausreichend sind, sollten demnach im Anschluss komplexere Methoden zur Optimierung des übergebenen Kontexts oder der Optimierung des LLMs angewendet werden. Prompt Engineering stellt aufgrund seiner Einfachheit und Flexibilität eine attraktive Alternative dar, insbesondere für Anwendungsfälle, in denen die hohen Anforderungen von LLMs an die Datenmenge die Anwendung von Fine-Tuning unmöglich machen. Im Hinblick auf den Mangel an qualitativ hochwertig annotierten Daten scheint es somit auch der geeignete Ansatz für das Argument Mining zu sein. Der Empfehlung von OpenAI (2024d) Prompt Engineering als ersten Ansatz für die Anwendung eines LLMs auf einen eigenen Anwendungsfall zu verwenden, wird gefolgt.

In der vorliegenden Untersuchung soll die Wirksamkeit der Methoden des Prompt Engineerings für den Anwendungsfall des Argument Minings untersucht werden. Der Einsatz vortrainierter Modelle ohne umfangreiche Anpassungen könnte den Aufwand für Argument Mining erheblich reduzieren. Trotz der Popularität großer Sprachmodelle gibt es nach meinem Kenntnisstand zum jetzigen Zeitpunkt nur einzelne Studien wie die von Yeginbergen et al. (2024), die sich explizit auf deren Anwendung für Argument Mining und die Rolle von Eingabeaufforderungen fokussieren. Hier soll diese Untersuchung einen Beitrag leisten. Die vorliegende Masterarbeit zielt darauf ab, die Potenziale und Herausforderungen von LLMs 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 für das Argument Mining zu verbessern. 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?

Nach (Peldszus & Stede, 2013, S. 6) können Argumente mit ihren Komponenten und Beziehungen in einem Argument-Graphen abgebildet werden. Demnach gibt es verschiedene Theorien zu den Strukturen von Argumenten mit zunehmender Komplexität (Peldszus & Stede, 2013, S. 3–14). Auch wenn eine visuelle Darstellung von Argumentationen die Nachvollziehbarkeit unterstützt, ist dies lediglich informativ aufgeführt und nicht Teil der vorliegenden Untersuchung. Zur Beantwortung der Forschungsfrage werden zunächst in Kapitel 2 das ausgewählte Modell, die verwendeten Daten sowie die Methode der Datenanalyse erläutert. Darauf aufbauend werden in Kapitel 3 die Ergebnisse der Untersuchung dargestellt, sodass diese in Kapitel 4 diskutiert und mögliche Handlungsempfehlungen abgeleitet werden können.

# Daten und Methoden

- Vorgehen Datengenerierung und -aufbereitung

- Datenlage und -qualität

- Vorgehen und Methoden der Datenanalyse

Kapitelbeschreibung

## Large Language Model

Es gibt mittlerweile eine Vielzahl von LLMS. Für die Untersuchung wird das Model GPT-4o mini von OpenAI verwendet. Dieses wird seitens OpenAI (2024b) als ihr kosteneffizientestes kleines Modell ausgewiesen. Nach den Angaben von OpenAI (2024b) übertrifft es in akademischen Benchmarks andere LLMs wie Gemini Flash, Claude Haiku und GPT-3.5 Turbo. Eine ausführliche Dokumentation bekräftigt die Entscheidung. Das ebenfalls von OpenAI in den gleichen Benchmarks besser abschneidende LLM GPT-4o wurde aufgrund der höheren Kosten und geringeren Anfragebegrenzungen (OpenAI, 2025c, 2025d) im Hinblick auf den begrenzten Bearbeitungszeitraum und das Budget nicht ausgewählt. Die Kosten von GPT-4o mini betragen zum derzeitigen Stand (01/2025) 0,15 USD pro einer Million Input-Tokens, wohingegen die Kosten für GPT-4o bei 2,5 USD pro einer Million Tokens liegen (OpenAI, 2025c) und somit etwa das 16,7-Fache höher sind.

**Tokens, Kontextfenster**

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 Tokens auf einmal verarbeiten können und sich andererseits die Kosten zur Verwendung des Modells GPT-4o mini nach der übergebenen Tokenanzahl richtet. Die Kosten pro Token sind für jedes Modell individuell. Darüber hinaus 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-Tokens repräsentieren die vom LLM generierten Antworten und Reasoning-Tokens werden von dem LLM bei der Generierung einer Antwort genutzt (OpenAI, 2024c). Die maximale Anzahl an Output-Tokens ist bei GPT-4o mini auf 16.384 Tokens begrenzt (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, 2025a). 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. Die restlichen Modellparameter wurden bei den Standardwerten belassen.

**Structured Output**

Nach der Beschreibung von OpenAI (2024c) akzeptiert GPT-4o mini sowohl Texte als auch Bilder als Eingaben und produziert Texte als Ausgabe. Hierbei unterstützt es strukturierte Ausgaben. Damit kann sichergestellt werden, dass die Ausgaben des LLMs dem übergebenen JSON-Schema entsprechen und sich auf die wesentlichen Informationen beschränken (OpenAI, 2024f). Die Ausgabe des LLMs in Form von Text als unstrukturierte Daten würde aus meiner Sicht die Weiterverarbeitung erschweren, weshalb die Auferlegung eines JSON-Schemas für eine semi-strukturierte Ausgabe als wesentlicher Vorteil angesehen wird. Für den vorliegenden Anwendungsfall wurde solch ein JSON-Schema eigenständig erstellt und bei den Anfragen an das LLM mit übergeben.

**Batch API**

Das Modell wird über die OpenAI Batch API verwendet. Dabei werden die Anfragen gesammelt übergeben und von OpenAI innerhalb von 24 Stunden bearbeitet (OpenAI, 2025b). Die Ausgaben des Modells inkl. dazugehöriger Metadaten können anschließend heruntergeladen werden. Aufgrund des Bearbeitungszeitraums von 24 Stunden gewährt OpenAI auf die Kosten einen Preisnachlass von 50% (OpenAI, 2025b).

## Datensatz

Um dem oben beschriebenen Mangel an annotierten Daten für das Argument Mining entgegenzuwirken, führen Lawrence & Reed (2020, S. 780, 798) an, dass sich einige Untersuchungen mit der Erstellung von Annotations-Richtlinien beschäftigen. Sie weisen jedoch auf den Nachteil hin, dass sich die spezifischen Annotations-Richtlinien auf die Besonderheiten des jeweiligen Bereichs beschränken, in dem sie entwickelt wurden und sich somit darauf aufbauende Methoden auch nur für diesen Bereich eignen (Lawrence & Reed, 2020, S. 806). Zudem kann es trotz dieser Richtlinien zu Abweichungen aufgrund subjektiver Einschätzungen kommen (Peldszus & Stede, 2013, S. 27). Die Übereinstimmung zwischen den Annotatoren kann als Gütemaß für die Zuverlässigkeit der Annotation herangezogen werden (Cabrio & Villata, 2018, S. 5428)*.* Es gibt verschiedene Datensätze, welche sich in ihrem Schwerpunkt und den Annotationen unterscheiden (Lawrence & Reed, 2020, S. 780–786). Cabrio & Villata (2018, S. 5432) bieten in ihrer Arbeit einen Vergleich von verfügbaren Datensätzen für das Argument Mining. Sie weisen in diesem Zusammenhang darauf hin, dass aufgrund fehlender eindeutiger Definitionen die Argumente in den Datensätze unterschiedlich annotiert werden und sich somit auch nicht für jede Teilaufgabe des Argument Minings eignen. Domänenunabhängige Rahmenbedingungen gibt es folglich nicht.

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 Argument 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 sowohl die Argumentationskomponenten als auch die argumentativen Beziehungen ausgewiesen werden. Zudem soll der Datensatz annotierter sein, damit anhand der Grundwahrheit die Ausgaben des LLMs evaluiert werden können. Sofern kein passender Datensatz verfügbar ist, bestünde die Möglichkeit einen eigenen Datensatz zu erstellen. Der Schwerpunkt der Untersuchung liegt jedoch auf der Anwendung von Eingabeaufforderungen in LLMs für Argument Mining und nicht auf der sprachwissenschaftlichen Theorie zu Argumenten. In Kombination mit dem Aufwand für die Datenbeschriftung wird sich gegen 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. Die Annotationen des Datensatzes weisen eine hohe Qualität auf, die mit der Erstellung eines eigenen Datensatzes voraussichtlich nicht hätte erreicht werden können. Der Datensatz besteht aus 402 von der Webseite essayforum.com zufällig ausgewählten überzeugenden Aufsätzen (Stab & Gurevych, 2017b, S. 630). Solche überzeugenden Aufsätze eignen sich nach Cabrio & Villata (2018, S. 5429)gut für das Argument Mining, da sie ein bestimmtes Thema erläutern wobei der Autor versucht die Leser von seinem Standpunkt zu überzeugen.

**Ergebnisse aus der EDA**

Der Datensatz enthält sowohl die vollständigen Aufsätze als Text-Dateien, als auch die Annotationen als ann-Dateien. Die Annotationen wurden von Stab & Gurevych (2017b, S. 630) unter Verwendung des *brat rapid annotation tools* erstellt. Bei den Annotationen werden die Argumentationskomponenten Hauptaussage (engl. MajorClaims), Behauptungen (engl. claims) und Prämissen (engl. 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 dafür (engl. for) oder dagegen (engl. against) markiert. Prämissen hingegen können eine Behauptung oder eine andere Prämisse unterstützen (engl. support) oder angreifen (engl. attack). Es ist möglich, dass es mehrere Hauptaussagen zu einem Text gibt. Hierbei wurde jedoch nicht annotiert, auf welche Hauptaussagen sich die Behauptungen beziehen. Die nachfolgende Abbildung dient zur Nachvollziehbarkeit der beschriebenen argumentative Struktur.

**Abbildung 1**   
Beispiel für argumentative Struktur der Aufsätze

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

Automatisch generierte Beschreibung

Eigene Darstellung nach Stab & Gurevych (2017b).

Ein Aufsatz umfasst ca. 200 bis 500 Wörter, mit einem Median von 319 Wörtern. Gemäß der dazugehörigen Annotationen enthalten die Aufsätze zwischen 1-3 Hauptaussagen, 2-10 Behauptungen, 2-20 Prämissen und 5-26 Beziehungen. Es wurde fälschlicherweise angenommen, dass die Aufsätze keine Duplikate enthalten, da es sich um einen professionell erstellten Datensatz handelt. Es wurde, nachdem die Anfragen bereits an das LLM gesendet wurden, jedoch festgestellt, dass ein Text dreimal und ein weiterer Text zweimal vorkommt. Die abweichenden Annotation dieser Texte resultiert vermutlich aus der zuvor beschriebenen subjektiven Einschätzung der Annotatoren. Glücklicherweise befindet sich nur einer dieser Texte im Trainingsdatensatz, weshalb die Prompts und somit auch die Anfragen an das LLM nicht überarbeitet werden mussten.[[1]](#footnote-1)

Yeginbergen et al. (2024, S. 11690) weisen darauf hin, dass beim 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.

## Methode

Im Rahmen der Untersuchung wird ein experimenteller Ansatz verfolgt. Anstatt ein Modell für jede Teilaufgabe des Argument Mining zu trainieren, soll sich die gute Performance der LLMs bei NLP-Aufgaben und deren Lernfähigkeit zu Nutze gemacht werden und es auf den vollständigen Prozess des Argument Minings angewendet werden. Dem LLM wird folglich ein Text aus dem Datensatz übergeben, aus welchem es die Argumentationskomponenten und deren Beziehungen extrahieren soll. Neben dem Texten werden dem Modell zusätzlich systematisch abweichende Prompts übergeben, sodass deren Auswirkungen auf die Leistung des LLMs analysiert werden kann. Bevor die Prompts an das LLM übergeben werden, werden die ann-Dateien aufbereitet. Die argumentativen Komponenten sind mit Tx versehen, wobei x eine fortlaufende Nummer ist. Es wird dabei folglich nicht zwischen den Argumentationskomponenten unterschieden. Diese IDs werden geändert, sodass anhand derer erkennbar ist, um welche Argumentationskomponente es sich handelt. Darauf aufbauen werden die annotierten Daten in ein JSON-Schema überführt. Das semi-strukturierte Format wird als geeignet betrachtet, da sowohl die Daten als auch die Ausgaben des LLMs, wie zuvor beschrieben, in dieses Format überführt werden können. Damit lassen sich die Daten für die Evaluation miteinander verglichen. Zudem können die Ergebnisse dann für jede Argumentationskomponente und die Beziehungen individuell betrachtet werden. Die argumentativen Beziehungen basieren auf den Argumentationskomponenten. In dem Prompt und der Grundwahrheit werden bei den Beziehungen für die Argumentationskomponenten die IDs anstelle der Texte verwendet, da somit weniger Tokens benötigt werden. Für die Evaluation werden die IDs anhand der dazugehörigen Textabschnitte ersetzt und in ein Tupel mit dem Schema (Ursprung, Art der Beziehung, Ziel) überführt. Entlang der Untersuchung werden die Daten unter Verwendung der Programmiersprache Python aufbereitet und die daraus resultierenden Prompts mittels spezieller Pakete wie der OpenAI Bibliothek an das LLMs übergeben.

### Prompts

Es gibt verschiedene Prompt Engineering Techniken. So kann beim ICL in Zero-Shot, One-Shot und Few-Shot unterschieden werden (Brown et al., 2020, S. 6–7; 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. Neben der Ergänzung von Beispielen in den Prompts gibt es noch weitere Ansätze. Dazu gehören beispielsweise Chain-of-thought prompting und die Verwendung einer Persona. Nachfolgend werden diese Techniken sowie deren Anwendung für die Untersuchung erläutert.

* **Zero-Shot 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 prompting (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 wurden zufällig ausgewählt.
* **Few-shot prompting (FS):** Auch hier werden wie beim OS dem LLM zusätzlich zur Aufgabenbeschreibung zufällige Beispiele als Input-Output Paare übergeben. Brown et al. (2020, S. 6, 10) verwenden dabei in der Regel zwischen 10 und 100 Beispiele, 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, da die Übergabe von zu vielen Beispielen kann zum Overfitting führen kann. Die Übergabe der Beispiele soll dazu führen, dass das LLM daraus Muster erkennt, die für die Bearbeitung der Aufgabe zuträglich sind (Ozdemir, 2024, S. 136; Yeginbergen et al., 2024, S. 11690). Mit Hinblick auf die Tokenanzahl und das Kontextfenster wurde die Anzahl der Beispiele stufenweise verdoppelt, begonnen bei 10 über 20 bis hin zu 40 Beispielen. Es wird das Ziel verfolgt einen groben Trend abzuleiten, anstatt eine optimale Anzahl an Beispielen zu ermitteln.
* **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. Auf den Anwendungsfall Argument Mining übersetzt werden dem Modell die Teilaufgaben genannt und beschrieben.
* **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). Für den vorliegenden Anwendungsfall wird dem LLM mitgeteilt dass es ein Experte für Argument Mining sei.

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. Diese Textbausteine enthalten die Aufgabenbeschreibung, das Ausgabeformat, die schrittweise Aufgabenbeschreibung und die Beschreibung der Persona. Diese Textbausteine werden dann um Beispiele ergänzt und miteinander kombiniert. Damit soll verhindert werden, dass leichte Abweichungen in der Formulierung die Ergebnisse verzerren. Darüber hinaus könnten anhand dessen flexibel weitere Prompts konstruiert werden. Die Eingabeaufforderungen sind in Englisch formuliert, da der Datensatz englische Texte beinhaltet und die multilingualen Fähigkeiten des LLMs nicht Teil dieser Untersuchung sind. Konkret werden dem LLM im Sinne des ZS, OS und FS eine Aufgabenbeschreibung mit 0, 1, 10, 20 und 40 Beispielen übergeben. Diese grundlegende Prompt-Struktur wird um die Textbausteine Persona oder COT oder beiden ergänzt. Hieraus ergeben sich insgesamt 20 verschiedene Prompts, anhand derer die Auswirkungen der Prompt Engineering Techniken analysiert werden können. Die Anzahl der übergebenen Tokens pro Prompt steigt mit zunehmender Komplexität. Der ZS-Prompt umfasst 82 Tokens, wohingegen der FS-Prompt mit 40 Beispielen, einer Persona und COT 54.470 Tokens groß ist.[[2]](#footnote-2) Da 40 Aufsätze als Beispiele verwendet werden und somit als Trainingsdaten zählen, können zur Evaluation abzüglich der Duplikate 359 Aufsätze als Testdatensatz herangezogen werden.

Um die Generalisierungsfähigkeit der Prompts bestmöglich bewerten zu können, wird jeder Prompt in Kombination mit jedem Text aus dem Testdatensatz an das LLM übergeben. Daraus ergeben sich 7180 Anfragen an das LLM. Diese Anfragen werden in sogenannten Batches gesammelt und entsprechend den Anfragebegrenzungen[[3]](#footnote-3) stückweise über die API an das LLM übergeben. Der vollständige Prozess von der Datenaufbereitung bis hin zu Evaluation kann schematisch in der Abbildung X im Anhang Y eingesehen werden.

OpenAI (2025e) unterscheidet bei der Übergabe von Nachrichten an das LLM verschiedene Rollen, welche Beeinflussen, wie das LLM die Eingabe interpretiert. Demnach können mit der Rolle User Anweisungen an das LLM übergeben werden, um eine Ausgabe zu erzeugen. Sie vergleichen es mit der Eingabe einer Nachricht bei ChatGPT. Mit der Rolle Developerkönnen ebenfalls Anweisungen an das Modell übergeben werden, jedoch haben sie Vorrang vor den Nachrichten der User-Rolle. Damit können die Ausgaben des Modells unabhängig von der Benutzereingabe beeinflusst werden. Die Anfragen an das LLM sind unter Berücksichtigung dieser Rollen so aufgebaut, dass der Prompt der Rolle Developerund der Aufsatz der Rolle Userzugewiesen sind. Damit soll das Szenario imitiert werden, dass ein Benutzer einen Text übergibt, aus dem die Argumentationskomponenten und deren Beziehungen extrahiert werden soll, wobei über die Developer-Rolle das Verhalten des LLMs gesteuert wird. Die Ausgaben des LLMs werden zur Evaluation der Leistung mit der Grundwahrheit abgeglichen. Hierzu sind geeignete Metriken heranzuziehen.

### Metriken zur Evaluation / Evaluationsmetriken

Bei der Wahl einer geeigneten Evaluationsmetrik gibt es aufgrund der unstrukturierten Art von Texten einige Besonderheiten, die es zu berücksichtigen gilt. So kann es vorkommen, dass die vom Modell extrahierten Textabschnitte von der Grundwahrheit, den Annotationen, abweichen können, indem mehr oder weniger Wörter einer Argumentationskomponente zugeordnet werden. Ein Textabschnitt, welcher nicht exakt mit der Grundwahrheit übereinstimmt, würde demnach als falsch gewertet werden. Dies ist jedoch eine strenge Definition, welche aufgelockert werden kann, indem man eine gewissen Grenze für die Übereinstimmung festlegt, ab welcher ein Text als übereinstimmen mit der Grundwahrheit gilt. Metriken, die auf der semantischer Ähnlichkeit beruhen werden nicht herangezogen, da die Argumentationskomponenten möglichst exakt und nicht sinngemäß extrahiert werden sollen. Als Metrik zur Berechnung der Übereinstimmung von zwei Textabschnitten wird BLEU(Bilingual Evaluation Understudy) herangezogen. Diese Metrik wurde von Papineni et al. (2002) zur Bewertung von maschinellen Übersetzungen entwickelt und hat sich dort nach (Chen & Cherry, 2014) als Standard etabliert. Sie kann jedoch auch auf ähnliche Aufgaben angewendet werden. BLEU basiert auf der Metrik Precision und misst, wie ähnlich ein generierter Text zu einem Referenztext ist. Laut der Beschreibung von Papineni et al. (2002) werden zur Bewertung der Übereinstimmungen n-Gramme herangezogen. Ein n-Gramm ist eine Folge von n aufeinanderfolgenden Elementen. Bezogen auf den vorliegenden Anwendungsfall sind die Elemente Wörter in einem Text. Dabei werden Wörter die häufiger in dem generierten Text, als in dem Referenztext vorkommen, sowie kurze generierte Texte bestraft. Damit soll sichergestellt werden, dass die Texte in Länge, Wortwahl und Reihenfolge der Wörter übereinstimmen. Der BLEU-Score kann zwischen 0 und 1 liegen. Je höher der Wert, desto größer die Übereinstimmung, mit dem Wert 1 bei einer identischen Übereinstimmung. Bei der Berechnung des BLEU-Scores besteht das Problem, dass wenn größere n-Gramme, wie bei n=4, für einen Text eine Precision von Null haben, der BLEU-Score für den Text ebenfalls Null ist, ungeachtet der Übereinstimmungen kleinerer n-Gramme, was wiederum zu einer verzerrten Bewertung führen kann (Chen & Cherry, 2014, S. 362). Chen & Cherry (2014, S. 362) haben deshalb sieben verschiedene Glättungsverfahren verglichen, die dieses Problem beheben. Die Methoden wurden hinsichtlich ihrer Korrelation mit menschlicher Beurteilung bewertet. Die Implementierung von BLEU inklusive der Glättungsfunktion erfolgt über die Python-Bibliothek NLTK. Es wird die erste Glättungsfunktion angewendet,[[4]](#footnote-4) bei der in Fällen ohne Übereinstimmung der Wert Null durch einen kleinen positiven Wert ersetzt wird, damit der BLEU-Score für den Text nicht ebenfalls Null wird. Die folgenden zwei Beispielsätze sollen einen Eindruck für den BLEU-Score vermitteln. Sie unterscheiden sich darin, dass a Großschreibung am Satzanfang und einen Punkt am Satzende besitzt, wohingegen dies bei b nicht der Fall ist.

1. Das ist ein Beispieltext für die Berechnung des BLEU-Scores.
2. das ist ein Beispieltext für die Berechnung des BLEU-Scores

Die Übereinstimmung dieser beiden Sätze entspricht einem BLEU-Score von ca. 0,76. Für die Festlegung einer Grenze, ab wann zwei Texte als übereinstimmend gelten wurde sich an diesem Beispiel orientiert und ein Grenzwert für den BLEU-Score von 0,75 festgelegt. Da die argumentativen Beziehungen auf den Argumentationskomponenten aufbauen, wird auch dabei der BLEU-Score angewendet. Das Tupel (Ursprung, Art der Beziehung, Ziel) gilt dann als korrekt, wenn es in allen Punkten übereinstimmt. Anhand dieser Festlegungen werden die Ausgaben des LLMs als korrekt oder falsch klassifizieren und in einer Konfusionsmatrix zusammengetragen. Bei einer Konfusionsmatrix handelt es sich um eine Tabelle zur Bewertung von Klassifikationen, welche die Anzahl der richtigen und falschen Klassifikationen anhand der vorhergesagten und der tatsächlichen Klasse abbildet (Bruce et al., 2020, S. 221). Nachfolgend werden die Felder einer Konfusionsmatrix an dem Beispiel von Behauptungen erläutert. Diese lassen sich sinngemäß auf die anderen Argumentationskomponenten übertragen.

* **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 es welche 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 werden folglich keine nicht-argumentativen Texte ausgegeben.

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. Im Hinblick auf den Aufwand und der Möglichkeit, dass LLMs Wörter hinzufügen können, die nicht Teil des Ursprungstextes sind, wird dieser Ansatz nicht weiterverfolgt.

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. Zum besseren Verständnis werden diese Metriken erneut an dem Beispiel der Argumentationskomponente Behauptung erläutert. Precision misst den Anteil der identifizierten Behauptungen, die tatsächlich Behauptungen sind. Recall ist hingegen der Anteil der tatsächlichen Behauptungen, die korrekt als solche extrahiert wurden. Eine hohe Precision allein könnte bedeuten, dass nur wenige tatsächliche Behauptungen vom LLM identifiziert werden. Der Fokus auf einen hohen Recall ohne ausreichende Precision könnte hingegen bedeuten, dass das LLM viele tatsächliche Behauptungen erkennt, aber auch viele Argumentationskomponenten fälschlicherweise als Behauptungen klassifiziert. Es ist somit sinnvoll ein ausgewogenes Verhältnis zwischen den beiden Metriken zur Bewertung der Gesamtleistung des LLMs heranzuziehen. Hierfür eignet sich der F1-Score, welcher Precision und Recall in einem Wert vereint, indem das harmonische Mittel aus ihnen gebildet wird (Géron, 2022, S. 111). Bei einem Wert von 1 trifft das Modell perfekte Vorhersagen, wohingegen ein Wert von 0 bedeutet, dass das Modell keine korrekten Vorhersagen macht. Die Ergebnisse pro Text werden anhand der Prompts gruppiert und gemittelt.

# Ergebnisse

- Möglichst objektive Darstellung der Ergebnisse (Ergebnisse nur beschreiben, nicht interpretieren)

- Idealerweise 1-3 zentrale Visualisierungen der Kernaussagen. Pro-Tipp: Aussagestarke Grafik in Text und ausführliche Tabelle im Anhang

Aus der zuvor beschriebenen Vorgehensweise resultieren die nachfolgenden Ergebnisse. Die Abbildung 2 stellt in vier Graphen den durchschnittlichen F1-Score der des LLMs für den Testdatensatz in Abhängigkeit der Tokenanzahl für die verschiedenen Prompts dar. Die Graphen sind nach Hauptaussagen, Behauptungen, Prämissen und Beziehungen unterteilt. Die Prompts sind darin in vier Gruppen eingeteilt. Die Gruppen gehen von der Aufgabenbeschreibung als Ausgangspunkt aus und richten sich danach, ob eine Persona, COT oder beides ergänzt wurden. Damit kann der Verlauf des F1-Scores in Abhängigkeit von der Anzahl der im Prompt übergebenen Beispiele für die verschiedenen Ansätze nachverfolgt werden. Der Ansatz, mit dem höchsten F1-Score in dem jeweiligen Graph wird durch eine blaue Markierung hervorgehoben. Die Aufgabenbeschreibung als Bezugsgröße ist hingegen in Schwarz/Grau dargestellt. Zur Orientierung für die Anzahl an übergebenen Beispielen sind vertikale Linien vorhanden.

**Abbildung 2**   
Durchschnittlicher F1-Score für Argumentationskomponenten und Beziehungen pro Prompt

Ein Bild, das Text, Diagramm, Plan, Zahl enthält.

Automatisch generierte Beschreibung

Eigene Darstellung.

Die Analyse der F1-Scores zeigt eine uneinheitliche Modellleistung. Vergleicht man zunächst die Graphen untereinander, so lässt sich erkennen, dass die Hauptaussagen den höchsten F1-Score erreichen, gefolgt von den Prämissen, dann den Behauptungen und abschließend den Beziehungen. Sowohl bei den Argumentationskomponenten als auch bei den Beziehungen erhöht sich der F1-Score mit zunehmender Tokenanzahl, was stellvertretend für die Anzahl an übergebenen Beispielen steht, deutlich. Der größte Anstieg erfolgt von keinem auf ein Beispiel. Bei der Erweiterung der Beispiele von 20 auf 40 ist es weniger deutlich

**Hauptaussagen:**

**Behauptungen**:

**Prämissen**:

**Beziehungen**:

stagniert der Wert oder fällt sogar ab.

Diskussion: Im Vergleich zur Aufgabenbeschreibung tragen die Ergänzung einer Persona, bei den Hauptaussagen und Prämissen in Verbindung mit COT, zu einer Erhöhung der Leistung bei. Dieser Vorteil reduziert sich jedoch oder entfällt mit zunehmender Anzahl an übergebenen Beispielen.

Die Übergabe von Beispielen verhält sich bis zu einer gewissen Anzahl ähnlich zum abnehmenden Grenznutzen (Gossen, 1854, S. 4–5)

%-Sätze der Steigerungen aufführen

Baseline bilden, bspw. Zero-Shot und anhand dessen den %-Leistungszuwachs bestimmen für die Hinzunahme von Beispielen und COT und Persona bzw. beides.

Als Bezugswert für den F1-Score wird der Prompt mit der niedrigsten Tokenanzahl herangezogen. Dies ist der ZS-Prompt, welcher lediglich die Aufgabenbeschreibung enthält. Abbildung 3 bildet die prozentuale Abweichung jedes Prompts von dem F1-Score des Bezugswertes ab.

**Abbildung 3**   
Abweichung der F1-Scores vom Bezugswert

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

Automatisch generierte Beschreibung

Eigene Darstellung.

Auch anhand dieser Abbildung wird schnell deutlich, dass Aufnahme von mindestens einem Beispiel in dem Prompt die Leistung deutlich verbessert. **ZS**: Bei den ZS-Prompts zeigt sich, dass die Ergänzung einer Persona zur Erhöhung des F1-Scores beiträgt, wohingegen die Ergänzung einer COT das Gegenteil bewirkt. Für die Beziehungen verschlechtert sich die Leistung dadurch soweit, dass selbst in Kombination mit der Persona keine Verbesserung erzielt werden kann. **OS**: Bei OS-Prompt lässt sich der negative Einfluss bei der Ergänzung von COT ebenfalls, wenn auch weniger deutlich, erkennen. Auffällig ist hierbei, dass bei den Hauptaussagen und Prämissen die Kombination aus einer Persona und COT zu der höchsten Steigerung des F1-Scores innerhalb der OS-Prompts führt.

**FS10**:

**FS20**:

**FS40**:

# Diskussion und Handlungsempfehlungen

- Diskussion der Ergebnisse

- Möglicherweise Ableitung von Handlungsempfehlungen

- Limitationen, weiterführende (neue) Forschungsfragen, etc.

**Diskussion der Ergebnisse**

***Unterscheidung für MajorClaims, Claims, Premisses und Relations.***

Die besonders niedrige Leistung des LLMs bei Vorhersage der Beziehungen lässt sich höchstwahrscheinlich damit erklären, dass diese auf der Identifikation der Argumentationskomponenten aufbaut. Sofern die Argumentationskomponenten nicht korrekt identifiziert werden, werden auch die dazugehörigen Beziehungen als nicht korrekt gewertet.

**Unterscheidung ZS, OS; FS**

Die Ergänzung von Beispielen in Prompt verhält sich ähnlich zum Grenznutzen. Übersetzt auf den Anwendungsfall bedeutet dies, dass jedes weitere Beispiel einen geringeren Leistungszuwachs beiträgt, als das Vorherige.

Anhand der FS40-Prompts wird sogar deutlich, dass die Leistung nicht zwangsläufig mit mehr Beispielen immer besser wird.

Die Übergabe der Aufgabenbeschreibung

***Unterscheidung für Basic, Persona, COT und Persona-COT***

Die Eingabeaufforderungen tragen zur Leistungsverbesserung bei. Allerdings gibt es keine eindeutige Prompt Engineering Technik, die über alle Argumentationskomponenten sowie die Beziehungen durchgehend die beste Leistung erzielt.

Die Einbindung von COT erzielt im Vergleich zum Basis-Prompt keinen Leistungssteigerung. Abgesehen davon wird jedoch deutlich, dass die Ergänzung der „speziellen Promptteile“ (Persona, Persona-COT) bei weniger übergebenen Beispielen eine Leistungssteigerung herbeiführt. Dies verringert sich jedoch mit zunehmender Anzahl der Beispiele oder verschwindet sogar ganz.

Es ist denkbar, dass Verwendung einer strukturierten Ausgabe des LLMs die Wirkung von COT verringert.

Unter Berücksichtigung der Tokenanzahl ist eine Abwägung zwischen benötigter Genauigkeit und den damit verbundenen Kosten zu treffen. Die

Bei gleicher Performance sollte der Prompt mit weniger Token verwendet werden.

Abgleich mit der Leistung von menschlichen Experten

Abgleich mit dem Ansatz aus Paper zum Datensatz.

Das in (Stab & Gurevych, 2017b, S. 646) vorgestellte Modell erreicht nahezu menschliche Performance mit 95,2 % der menschlichen Leistung bei der Identifizierung von Komponenten, 87,9 % bei der Identifizierung von Beziehungen und 80,5 % bei der Erkennung von Haltungen.

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

Automatisch generierte Beschreibung

(Stab & Gurevych, 2017b, S. 648)

*„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).

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

Auch wenn es keine bessere Leistung hat, ist es wesentlich leichter umzusetzen, da kein Expertenwissen benötigt wird und damit auch günstiger.

Die Ergänzung von Beispielen und Persona teils in Verbindung mit COT besonders effektiv für die Verbesserung der

Ein effizienter Umgang mit der Tokenanzahl ein wichtiger Aspekt bei der Anwendung von LLMs. Je mehr Beispiele ein Prompt umfasst, desto größer die Tokenanzahl und umso höher sind auch die Kosten des Prompts. Hier sollte individuell eine Abwägung zwischen den Kosten und dem Nutzen erfolgen.

Der optimale Prompt hängt somit davon ab, welches Ziel oberste Priorität hat, höchste Genauigkeit oder Einhaltung einer Kostenobergrenze.

**Praktischer Mehrwert**

Die vorliegende Untersuchung gibt Aufschluss darüber, wie leistungsfähig LLMs für das Argument Mining sind und welche der betrachteten Prompt Engineering Techniken in diesem Zusammenhang die besten Ergebnisse liefern. Die Untersuchung trägt damit sowohl zur Weiterentwicklung der Forschung auf dem Gebiet des Argument Minings als auch zur praktischen Anwendung von LLMs in realen Anwendungsfällen bei.

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 das untersuchten LLM
* 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. 🡪Nicht domänenunabhängig anwendbar. Erweiterung des Datensatzes auf verschiedene Domänen. Ggf. ist die Datensatzgröße dann für ein Fine-Tuning ausreichend.
* Änhlichkeitsmetrik zum vergleich der Sätze (BLEU) berücksichtigt keine semantische Änhlichkeit. D.h. wenn das Modell singemäß die Aussgae korrekt extrahiert hat, diese aber nicht wie im Text formuliert ist, erhält sie einen niedrigen BLEU Score.

**Handlungsempfehlungen**

Die Untersuchung liefert einen ersten Ansatz für die Anwendung von LLMs für Argument Mining. Auch wenn die Einschätzung, ab wann die Leistung des Modells ausreichend ist, individuell vom konkreten Anwendungsfall ist, erscheint die bisher erzielte Leistung als noch nicht ausreichend für eine Verwendung in einer realen Anwendung. Besonders die Identifikation der Beziehungen zwischen den Argumentationskomponenten hat großen Verbesserungsbedarf. Auf Grundlage dieser Ergebnisse können jedoch weitere Untersuchungen anschließen.

Im Sinne der Optimierung des LLMs und unter der Voraussetzung, dass ein ausreichend großer Datensatz vorliegt, könnte ein Fine-Tuning des LLM vorgenommen werden. Im Sinne der Kontextoptimierung könnte hingegen die Einbindung von externen Quellen implementiert werden, um dem LLM relevante Informationen wie beispielsweise die Annotationsrichtlinien zur Verfügung zu stellen. Sollten diese Ansätze nicht möglich sein, ist die Untersuchung weiterer, leistungsstärkerer LLMs ebenfalls ein interessanter Schritt.

Sofern das LLM dann eine ausreichende Performance erzielt, wäre es interessant darauf aufbauend den Ansatz so weiterzuentwickeln, dass die Argumentationsstrukturen in ein Strukturdiagrammen überführt werden, um die extrahierte Argumentation visuell leicht verständlich aufzuarbeiten. Auch die Betrachtung für weitere Sprachen ist denkbar.

* 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

Bruce, P. C., Bruce, A., & Gedeck, P. (2020). *Practical statistics for data scientists: 50+ essential concepts using R and Python* (2. Aufl.). O’Reilly Media, Inc.

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

Chen, B., & Cherry, C. (2014). A Systematic Comparison of Smoothing Techniques for Sentence-Level BLEU. *Proceedings of the Ninth Workshop on Statistical Machine Translation*, 362–367. https://doi.org/10.3115/v1/W14-3346

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

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

Gossen, H. H. (1854). *Entwickelung der Gesetze des menschlichen Verkehrs, und der daraus fließenden Regeln für menschliches Handeln*.

Han, S., Wang, M., Zhang, J., Li, D., & Duan, J. (2024). A Review of Large Language Models: Fundamental Architectures, Key Technological Evolutions, Interdisciplinary Technologies Integration, Optimization and Compression Techniques, Applications, and Challenges. *Electronics*, *13*(24), 5040. https://doi.org/10.3390/electronics13245040

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

OpenAI. (2025b). *Batch API*. OpenAI Platform. https://platform.openai.com/docs/guides/batch/batch-api

OpenAI. (2025c). *Pricing*. https://openai.com/api/pricing/

OpenAI. (2025d). *Rate limits*. OpenAI Platform. https://platform.openai.com/docs/guides/rate-limits?context=tier-free

OpenAI. (2025e). *Text generation*. https://platform.openai.com/docs/guides/text-generation

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.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL ’02*, 311–318. https://doi.org/10.3115/1073083.1073135

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 beschriebenen Textbausteinen und zum anderen die TXT-Dateien.

**Abbildung 4**   
Zero-Shot Prompt-Struktur

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Eigene Darstellung.

**Abbildung 5**   
One-Shot Prompt-Struktur

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Eigene Darstellung.

Die Few-Shot Prompts sind systematisch gleich zu den One-Shot Prompts aufgebaut, jedoch werden mehrere Input-Output-Paare übergeben.

1. Eine detailliertere Beschreibung zum Umgang mit den Duplikaten kann aus dem 3.Notebook unter dem Abschnitt „Behandlung von Duplikaten“ entnommen werden. [↑](#footnote-ref-1)
2. Die Tabelle mit der Tokenanzahl pro Prompt befindet kann im Anhang XX eingesehen werden. [↑](#footnote-ref-2)
3. Die Anfragebegrenzungen richten sich bei OpenAI nach Stufen. Je höher die Stufe, desto höher die Anfragebegrenzung. Um in die nächsthöhere Stufe zu gelangen, müssen gewisse Voraussetzungen erfüllt sein. Für mehr Informationen siehe OpenAI (2025d). [↑](#footnote-ref-3)
4. Zwar bieten nach Chen & Cherry (2014, S. 364) die anderen Glättungsfunktionen eine höhere Korrelation, diese werden jedoch als marginal betrachtet. Es wird deshalb die Einfachheit dieser Methode bevorzugt wird. Zudem erreichen die Glättungsfunktionen 5 und 7 bei einer perfekten Übereinstimmung einen Wert > 1. Der BLEU-Score kann jedoch nur zwischen 0 und 1 liegen. Vermutlich ist die Implementierung in NLTK fehlerhaft. [↑](#footnote-ref-4)