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 Frank Fels
Email	benjamin.fels@student.dbuas.de
Matrikelnummer:	190200
Fachsemester:	5
Studiengang:	Data Science & Management
Datum Abgabe:	20.02.2025
Gutachter:	Prof. Dr. Marcel Hebing

Executive Summary

Argumentationen sind ein zentraler Bestandteil unserer täglichen Kommunikation. 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. Argumente bestehen aus den Komponenten Behauptung und Prämisse, die in einer strukturierten Beziehung zueinander stehen. Der Forschungsbereich, der sich mit der automatischen Extraktion dieser Argumentationskomponenten und ihrer Beziehungen beschäftigt, wird als Argument Mining bezeichnet. Der Fortschritt in der Entwicklung großer Sprachmodelle (Large Language Models, LLMs) eröffnet neue Möglichkeiten für das Argument Mining. Klassische regelbasierte oder maschinelle Lernansätze sowie das Trainieren eines speziellen LLMs sind oft ressourcenintensiv und erfordern große Mengen annotierter Trainingsdaten. LLMs bieten durch ihre Generalisierungsfähigkeit eine vielversprechende Alternative, insbesondere in Kombination mit Prompt Engineering, einem Prozess zur gezielten Steuerung von Modellausgaben durch optimierte Eingabeaufforderungen. Es stellt sich die Frage, inwiefern LLMs durch gezieltes Prompt Engineering für das Argument Mining optimiert werden können. Um diese Frage zu beantworten, wird GPT-4 mini von OpenAI als LLM herangezogen, wobei verschiedene Prompt Engineering Techniken getestet werden, um ihre Auswirkungen auf das Argument Mining zu bewerten. Es werden insbesondere der Einfluss von Beispielen, Chain-of-Thought (COT), welche die Teilaufgaben des Argument Mining vorgibt, und die Integration einer Persona untersucht. Insgesamt werden 20 verschiedene Prompt-Varianten getestet. Als Datensatz wird der Argument Annotated Essays Corpus von Stab und Gurevych (2017a) verwendet, welcher 402 überzeugende Aufsätze mit Hauptaussagen, Behauptungen und Prämissen sowie deren argumentativen Beziehungen enthält. Bis zu 40 Aufsätze werden in die Prompts integriert. Die restlichen Aufsätze werden zur Evaluation der Modellleistung verwendet. In Kombination mit den Prompt-Variationen ergeben sich daraus insgesamt 7180 Anfragen, die an das LLM gestellt und anschließend analysiert werden.

Die Untersuchung zeigt, dass der Einsatz von Techniken des Prompt Engineerings zu signifikanten Verbesserungen in der Identifikation und Strukturierung von Argumenten durch LLMs führt. Im Vergleich zu einer einfachen Aufgabenbeschreibung

kann mit der Anwendung von Prompt Engineering Techniken der F1-Score um bis zu 12 % für Beziehungen, 22 % für Behauptungen, 26 % für Prämissen sowie 41 % für Hauptaussagen erhöht werden. Die Anzahl der bereitgestellten Beispiele in einem Prompt hat eine Wirkung, die vergleichbar ist mit dem Gesetz vom abnehmenden Grenznutzen. Der größte Leistungszuwachs erfolgt demnach bereits mit der Bereitstellung eines einzelnen Beispiels. Eine weitere Erhöhung der Beispiele über 10 hinaus zeigt im Vergleich nur marginale Verbesserungen und ab 40 Beispielen lassen sich Anzeichen für Overfitting erkennen. Die Verwendung einer Persona innerhalb des Prompts verbessert die Modellleistung, wohingegen COT in vielen Fällen keine Verbesserung gezeigt hat und sogar zu Leistungseinbußen führen kann. Dies könnte an dem strukturierten Ausgabeformat liegen, welches dem LLM auferlegt wurde. In bestimmten Fällen kann die Kombination von Persona und COT jedoch Synergieeffekte erzeugen.

Trotz erheblicher Verbesserungen durch Prompt Engineering ist die Performance des LLMs im Vergleich zu den bisherigen Ansätzen des Argument Mining geringer, weshalb es noch nicht als verlässliche Alternative zu den spezialisierten Modellen angesehen werden kann. Dies unterstreicht die Notwendigkeit weiterer Optimierungsmaßnahmen. Nichtsdestotrotz verdeutlicht die Untersuchung das Potenzial von Prompt Engineering zur Steuerung der Generalisierungsfähigkeit von LLMs im Argument Mining. Basierend auf diesen Erkenntnissen ergeben sich mehrere Handlungsempfehlungen: Erstens sollte bei der Anwendung von LLMs im Argument Mining eine moderate Anzahl an Beispielen im Prompt verwendet werden, um Overfitting und unnötig hohe Token-Kosten zu vermeiden. Zweitens sollten Prompts mit einer Persona bevorzugt werden. Drittens könnten weiterführende Untersuchungen durch Fine-Tuning des LLMs oder die Integration externer Wissensquellen, etwa Annotationsrichtlinien, die Leistung weiter steigern. Auch eine dynamische Auswahl der Beispiele in Abhängigkeit vom Eingabetext oder die Ergänzung von strukturellen sowie kontextuellen Informationen könnte vielversprechend sein. Langfristig wäre zudem eine Erweiterung auf andere Sprachen und Domänen sinnvoll, um die Generalisierbarkeit der Ergebnisse zu erhöhen. Schließlich könnte die visuelle Aufbereitung der extrahierten Argumentationsstrukturen in Strukturdiagrammen den praktischen Nutzen der gewonnenen Erkenntnisse verbessern, indem sie die Nachvollziehbarkeit der Argumentationslogik erhöht.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

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

1 Einleitung

Argumente sind ein wichtiger Bestandteil in der menschlichen Kommunikation. Peldszus und 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 und Stede (2013) sowie Stab und Gurevych (2017b, S. 620) 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 und 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.

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. Zur Vereinfachung werden nachfolgend die Argumentationskomponenten und deren Beziehungen unter dem Begriff Argumentationsstruktur zusammengefasst. 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). 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 und 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 und 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 und 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 Modell 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, dem 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, dass mittels sogenannter Prompts ein LLM anhand von Beispielen an die gewünschte Aufgabe angepasst werden kann. Ein Prompt kann als ein Eingabetext verstanden werden, auf den 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 und dem maximal zulässigen Wert des Kontextfensters des LLMs gewählt werden, was typischerweise 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 und 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 Energie, das Fachpersonal und die Infrastruktur zur Verwendung des LLMs hinzu. Dieser Ansatz ist folglich unbezahlbar für eine Vielzahl von Forschenden[1] (Patil & Gudivada, 2024, S. 31). Die Anpassung eines bereits vortrainierten LLMs mittels Fine-Tuning für die eigene Anwendung ist

[1] Aus Gründen der besseren Lesbarkeit wird auf die gleichzeitige Verwendung der Sprachformen männlich, weiblich und divers verzichtet. Sämtliche Personenbezeichnungen gelten gleichermaßen für alle Geschlechter.

hingegen nach Patil und Gudivada (2024, S. 18) kostengünstiger. Brown et al. (2020, S. 6) führen jedoch an, dass für das Fine-Tuning ein Datensatz mit einem Umfang von typischerweise tausenden bis hunderttausenden 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 und 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 promptbasierte 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 wurde, 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 (o. J.-e) betont, dass Prompt Engineering bei der Anwendung von LLMs auf eigene Anwendungsfälle als erster 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 (o. J.-e), Prompt Engineering als ersten Ansatz für die Anwendung eines LLMs auf einen eigenen Anwendungsfall zu verwenden, wird gefolgt. Trotz der Popularität großer Sprachmodelle gibt es nach meinem Kenntnisstand zum jetzigen Zeitpunkt nur einzelne Arbeitspapiere, wie das von Cabessa et al. (2024), welche sich explizit mit deren Anwendung für Argument Mining und der Rolle von Eingabeaufforderungen auseinandersetzen. Es scheint somit ein wenig untersuchtes Forschungsgebiet zu sein. Cabessa et al. (2024) beschränken sich auf die Teilaufgabe der Klassifizierung der Argumentationskomponenten und verwenden dabei strukturelle und kontextuelle Informationen. Die vorliegende Untersuchung soll einen Beitrag zu diesem Forschungsgebiet liefern. Sie zielt darauf ab, die Potenziale und Herausforderungen von LLMs im Kontext des Argument Minings zu erforschen. Der Fokus liegt dabei auf der Anwendung von Techniken des Prompt Engineerings, um die Generalisierungsfähigkeiten dieser Modelle gezielt zu steuern und deren Leistung ohne Fine-Tuning zu verbessern. Die zentrale Forschungsfrage lautet somit: Wie beeinflusst der Einsatz von Prompt Engineering Techniken die Leistung von Large Language Models bei der automatisierten Erkennung von Argumentationskomponenten und deren Beziehungen?

Nach Peldszus und 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.

2 Daten und Methoden

Das Kapitel beginnt mit der Vorstellung des verwendeten LLMs und den Besonderheiten bei dessen Verwendung, wie der Batch API oder den Ansätzen zur Reproduzierbarkeit. Daran anschließend erfolgt die Begründung zur Auswahl des Datensatzes sowie die Beschreibung desgleichen. In dem Unterkapitel Methode wird die Vorgehensweise erläutert. Speziell wird hierbei auf die verwendeten Prompt Engineering Techniken und die Evaluationsmetriken eingegangen.

2.1 Large Language Model

Es gibt mittlerweile eine Vielzahl von LLMs. Für die Untersuchung wird das Modell GPT-4o mini von OpenAI verwendet. Dieses wird seitens OpenAI (2024) als ihr kosteneffizientestes kleines Modell ausgewiesen. Nach den Angaben von OpenAI (2024) 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, o. J.-f, o. J.-h) 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, o. J.-f) und somit etwa das 16,7-Fache höher sind.

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 der verwendeten Kodierung (engl. encoding) und kann 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 begrenzte Anzahl an Tokens auf einmal verarbeiten können und sich andererseits die Kosten zur Verwendung des Modells GPT-4o mini nach der übergebenen Tokenanzahl richten. 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, 2024). OpenAI (o. J.-d) definiert 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 die Output-Tokens sowie die 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, o. J.-d). Die maximale Anzahl an Output-Tokens ist bei GPT-4o mini auf 16.384 Tokens begrenzt (OpenAI, o. J.-d).

Die Ausgaben eines LLMs können standardmäßig bei gleicher Anfrage unterschiedlich ausfallen (Anadkat, 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, o. J.-a). Der Parameter *temperature* kann zwischen 0 und 1 festgelegt werden, wobei die Zufälligkeit der Ausgaben des LLMs mit steigendem Wert zunimmt (OpenAI, o. J.-c). 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, 2023). Diese Kennung kann sich bei notwendigen 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.

Nach der Beschreibung von OpenAI (o. J.-d) 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, o. J.-i). 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. Das Modell wird über die OpenAI Batch API verwendet. Dabei werden die Anfragen gesammelt übergeben und von OpenAI innerhalb von 24 Stunden bearbeitet (OpenAI, o. J.-b). 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, o. J.-b).

2.2 Datensatz

Um dem oben beschriebenen Mangel an annotierten Daten für das Argument Mining entgegenzuwirken, führen Lawrence und Reed (2020, S. 780, 798) an, dass sich einige Untersuchungen mit der Erstellung von Annotationsrichtlinien beschäftigen. Sie weisen jedoch auf den Nachteil hin, dass sich die spezifischen Annotationsrichtlinien 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 und 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 Datensätzen unterschiedlich annotiert werden und sich somit auch nicht für jede Teilaufgabe des Argument Minings eignen. Domänenunabhängige Rahmenbedingungen gibt es folglich nicht.

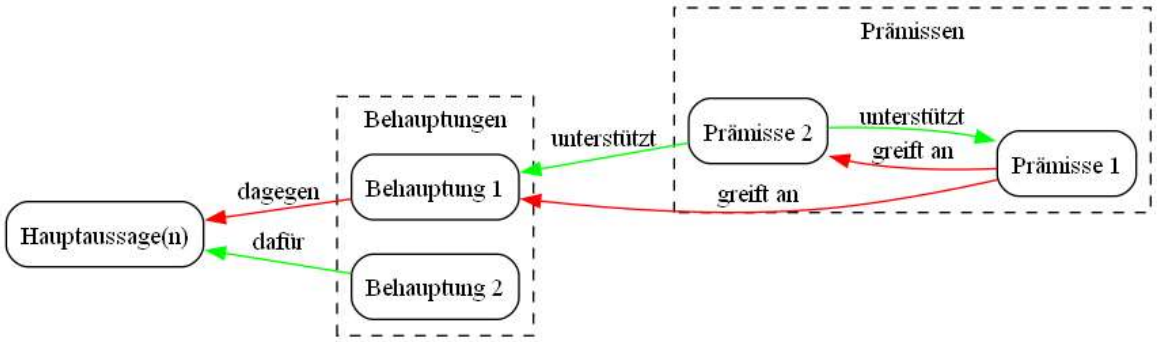
Für die Auswahl eines geeigneten Datensatzes wurden diverse Kriterien herangezogen. Zunächst sollte der Datensatz vorab nicht bereits von dem nicht-argumentativen Text befreit worden sein, um die Realität bestmöglich abzubilden (Stab & Gurevych, 2017b, S. 620). Des Weiteren wird der Ansatz verfolgt, 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 Annotationsrichtlinien von Stab und 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 und 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 und 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.

Der Datensatz enthält sowohl die vollständigen Aufsätze als Textdateien als auch die Annotationen als ann-Dateien. Die Annotationen wurden von Stab und Gurevych (2017b, S. 630) unter Verwendung des *brat rapid annotation tools* erstellt. Bei den Annotationen werden die Argumentationskomponenten Hauptaussage (engl. major claims), Behauptungen (engl. claims) und Prämissen (engl. premises) unterschieden. Nach Stab und Gurevych (2017b, S. 627) beinhalten solche Aufsätze folgende argumentative Struktur: Eine Hauptaussage spiegelt den Standpunkt des Autors wider, wobei dieser anhand von Argumenten unterstützt oder angegriffen wird. Ein Argument besteht aus einer Behauptung und mindestens einer Prämisse. Die argumentative Struktur der Aufsätze stimmt somit abgesehen von der Ergänzung einer Hauptaussage mit der oben beschriebenen allgemeinen Struktur von Argumenten überein. 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 argumentativen Struktur.

Abbildung 1
Beispiel für argumentative Struktur der Aufsätze



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 worden waren, jedoch festgestellt, dass ein Text dreimal und ein weiterer Text zweimal vorkommt. Die abweichende 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.[2]

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

[2] Eine detailliertere Beschreibung zum Umgang mit den Duplikaten kann aus dem 3.Notebook unter dem Abschnitt „Behandlung von Duplikaten“ entnommen werden.

enthalten allerdings auch nicht-argumentative Textstellen, welche keine Argumentationskomponenten darstellen.

2.3 Methode

Im Rahmen der Untersuchung wird ein experimenteller Ansatz verfolgt. Anstatt ein Modell für jede Teilaufgabe des Argument Minings zu trainieren, soll sich die gute Performance der LLMs bei NLP-Aufgaben und deren Lernfähigkeit zunutze gemacht werden und es soll auf den vollständigen Prozess des Argument Minings angewendet werden. Dem LLM wird folglich ein Text aus dem Datensatz übergeben, aus welchem es die Argumentationsstruktur extrahieren soll. Neben den Texten werden dem Modell zusätzlich systematisch abweichende Prompts übergeben, sodass deren Auswirkungen auf die Leistung des LLMs analysiert werden können. 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 aufbauend 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 vergleichen. 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 und spezieller Bibliotheken aufbereitet. Die daraus resultierenden Prompts werden mittels der OpenAI-Bibliothek an das LLM übergeben.

2.3.1 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 wird im 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 übergebenem Input und gewünschtem Output aufgestellt. Der Input ist in diesem Fall der Text des Aufsatzes und der Output die Argumentationsstruktur, formatiert 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 Ergebnissen. So weisen Google (2024) darauf hin, dass Experimente notwendig sind, um die optimale Anzahl an Beispielen zu bestimmen, da die Übergabe von zu vielen Beispielen 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, beginnend 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, o. J.-g; 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 (o. J.-g) 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, o. J.), 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 beide ergänzt. Hieraus ergeben sich insgesamt 20 verschiedene Prompts, anhand derer die Auswirkungen der Prompt Engineering Techniken analysiert werden können. Die durchschnittliche Tokenanzahl für die Aufsätze beträgt 372 Tokens und

915 Tokens für die als JSON-Objekte transformierten Annotationen. Ein einzelnes Input-Output-Paar umfasst demnach im Durchschnitt 1.287 Tokens. 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.[3] 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[4] stückweise über die API an das LLM übergeben. Der vollständige Prozess von der Datenaufbereitung bis hin zur Evaluation kann schematisch in der Abbildung 6 im Anhang eingesehen werden.

OpenAI (o. J.-j) 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 Developer können 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 Developer und der Aufsatz der Rolle User zugewiesen sind. Damit soll das Szenario imitiert werden, dass ein Benutzer einen Text übergibt, aus dem die Argumentationsstruktur 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.

[3] Die Tabelle mit der Tokenanzahl pro Prompt kann im Anhang eingesehen werden.
[4] 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 (o. J.-h).

2.3.2 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 gewisse Grenze für die Übereinstimmung festlegt, ab welcher ein Text als übereinstimmend mit der Grundwahrheit gilt. Metriken, die auf der semantischen Ähnlichkeit beruhen, werden nicht herangezogen, da die Argumentationsstruktur möglichst exakt und nicht sinngemäß extrahiert werden soll. 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 und 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 und 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,[5] 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.

a. Das ist ein Beispieltext für die Berechnung des BLEU-Scores.
b. 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 klassifiziert 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.

[5] Zwar bieten nach Chen und Cherry (2014, S. 364) die anderen Glättungsfunktionen eine höhere Korrelation, diese werden jedoch als marginal betrachtet. Es wird deshalb die Einfachheit dieser Methode bevorzugt. 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 der NLTK-Bibliothek fehlerhaft.

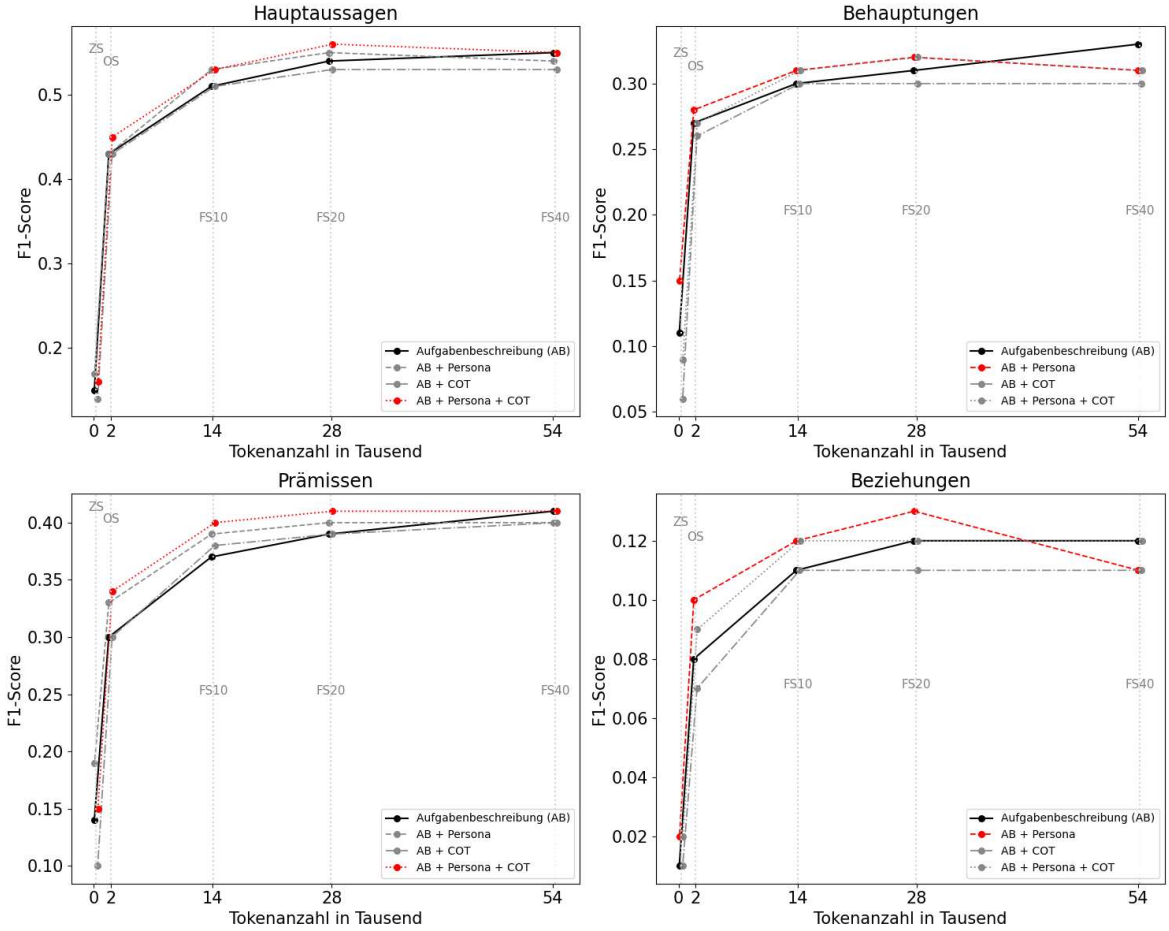
- **True Negative (TN)**: Die Textabschnitte werden korrekt nicht als Behauptung erkannt. Die Besonderheit bei dem vorliegenden Anwendungsfall ist, dass das LLM nur die argumentativen Texte extrahieren soll. Es werden folglich keine nicht-argumentativen Texte ausgegeben.

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 die Möglichkeit, dass LLMs Wörter hinzufügen können, die nicht Teil des Ursprungstextes sind, und den damit verbundenen Aufwand wird dieser Ansatz nicht weiterverfolgt. Nichtsdestotrotz lassen sich aus den Werten der Konfusionsmatrix diejenigen Klassifikationsmetriken berechnen, für welche die Anzahl der FN nicht benötigt wird. Folglich 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.

3 Ergebnisse

Aus der in Kapitel 2 beschriebenen Vorgehensweise resultieren die nachfolgenden Ergebnisse. Die Abbildung 2 stellt in vier Graphen den durchschnittlichen F1-Score des LLMs für den Testdatensatz in Abhängigkeit von 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 wurde. 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, welcher entlang dieser fünf Beispiel-Stufen in den meisten Fällen den höchsten F1-Score erreicht, wird in dem jeweiligen Graphen durch eine rote Markierung hervorgehoben. Die Aufgabenbeschreibung als Bezugsgröße ist hingegen in Schwarz dargestellt. Zur Orientierung, hinter welcher Tokenanzahl sich welche Anzahl an Beispielen verbirgt, sind vertikale Linien vorhanden. Diese richten sich nach der durchschnittlichen Tokenanzahl pro Stufe. Die Hauptaussagen, Behauptungen, Prämissen und Beziehungen kommen in dieser Reihenfolge zunehmend häufiger in den Aufsätzen vor. Ein LLM versucht ein Muster anhand der Beispiele aus den Trainingsdaten zu erkennen. Es wäre folglich anzunehmen, dass das LLM für diejenigen Komponenten die beste Leistung erzielt, für die es die meisten Einzelbeispiele erhält. Dem ist jedoch nicht so, wie in der Abbildung zu erkennen. Vielmehr verhält es sich fast gegenteilig.

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



Eigene Darstellung.

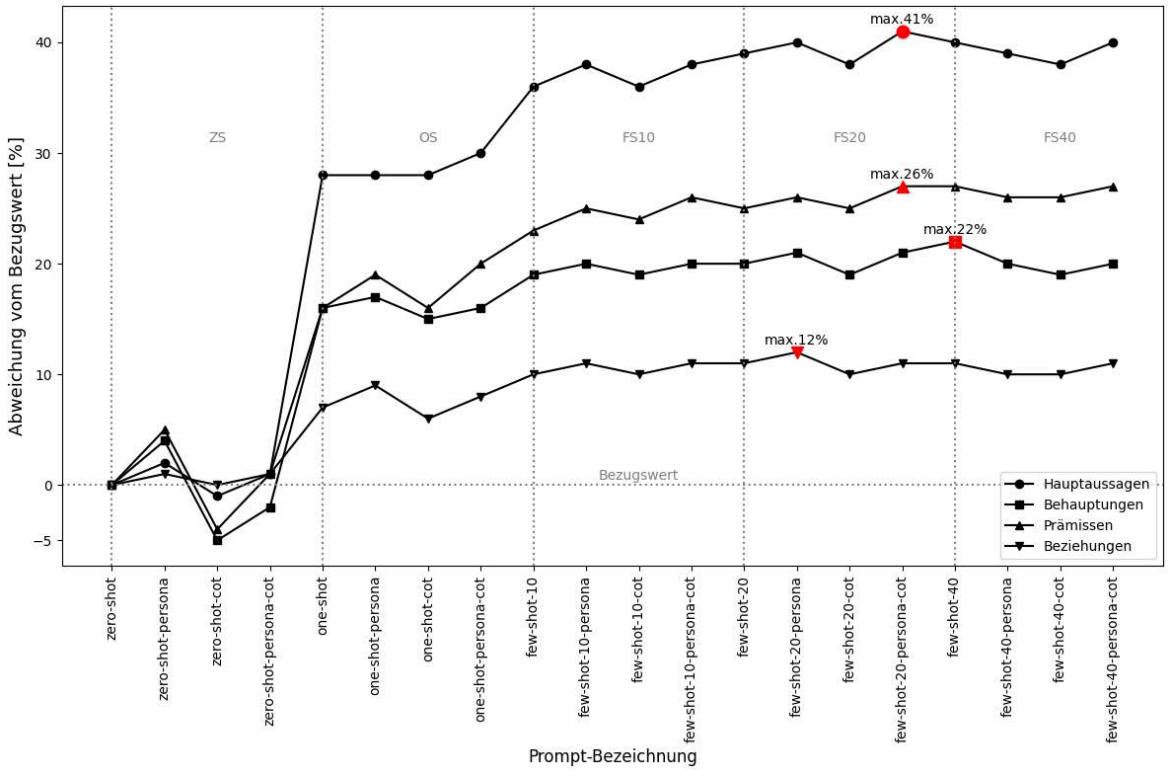
Vergleicht man die Graphen untereinander, so lässt sich erkennen, dass die Hauptaussagen den höchsten F1-Score mit 0,56 erreichen, gefolgt von den Prämissen mit 0,41, dann den Behauptungen mit 0,33 und abschließend den Beziehungen mit 0,13. Sowohl bei den Argumentationskomponenten als auch bei den Beziehungen 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. Für die Behauptungen und Beziehungen stagniert der F1-Score vereinzelt bereits bei der Erhöhung von 10 auf 20 Beispiele. In dem Schritt von 20 zu 40 Beispielen kommt es sowohl bei den Argumentationskomponenten als auch bei den Beziehungen zusätzlich vor, dass sich der F1-Score verringert.

Es lässt sich erkennen, dass die Ergänzung einer Persona in der Aufgabenbeschreibung in allen vier Betrachtungen zu einer Erhöhung des F1-Scores führt, bis zu einer Steigerung der Beispiele auf 20 Stück. Die Ergänzung von COT in der

Aufgabenbeschreibung ohne eine Persona führt hingegen oftmals zu einer Verringerung. Die Prompts, in welchen die beiden Ansätze kombiniert wurden, erreichen hingegen bei den Hauptaussagen und Prämissen in den meisten Fällen den höchsten F1-Score. Dieser Vorteil entfällt jedoch bei der Erhöhung auf 40 Beispiele. Die Prompts mit lediglich der Aufgabenbeschreibung erreichen in diesen Fällen einen gleichen oder sogar höheren F1-Score als diejenigen Prompts, in denen Ergänzungen vorgenommen wurden. Bildet man nun ein Verhältnis aus dem F1-Score zu der Tokenanzahl pro Prompt, dann ergibt sich ein gegenteiliges Bild. Damit sind die ZS-Prompts mit lediglich der Aufgabenbeschreibung oder in Kombination mit der Persona den restlichen Prompts überlegen.

Für eine weitere Betrachtung wird als Bezugswert für den F1-Score 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 dieses Bezugswertes ab.

Abbildung 3
Abweichung der F1-Scores vom Bezugswert



Eigene Darstellung.

Auch hier verdeutlicht die Analyse des F1-Scores eine uneinheitliche Modellleistung entlang der Argumentationskomponenten und Beziehungen. Der Einfluss der

Variationen der Prompts auf die Leistung des LLMs ist für die Beziehungen durchweg am niedrigsten. Auch hier zeigt sich, dass die Aufnahme von mindestens einem Beispiel in den Prompt die Leistung deutlich verbessert. Lediglich ein einzelnes Beispiel steigert die Leistung von 7 % für Beziehung bis hin zu 28 % für Hauptaussagen. Die maximale Leistungssteigerung beträgt für Beziehungen 12 %, für Behauptungen 22 %, für Prämissen 26 % und für Hauptaussagen 41 %. Drei von vier Prompts, die diese Leistungen erzielen, liegen in dem Bereich der FS20-Prompts. Bei den ZS-Prompts führt die Verwendung von COT zu einer Verringerung von bis zu 5 % bei den Behauptungen. Die Ergänzung einer Persona trägt hingegen zu einer Erhöhung des F1-Scores bei. Für die Beziehungen verschlechtert sich die Leistung durch COT jedoch so weit, dass selbst in Kombination mit der Persona keine Verbesserung im Vergleich zum Bezugswert erzielt werden kann. Im Bereich der OS-Prompts 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 größten Steigerung des F1-Scores innerhalb der OS-Prompts führt. Innerhalb der FS10-Prompts verändert COT, mit Ausnahme der Prämissen, die Leistung nicht. Die Ergänzung von Persona in der Aufgabenbeschreibung führt hingegen zu einer leichten Verbesserung um ein bis zwei Prozentpunkte. Entlang der FS20-Prompts verringert COT und verbessert Persona den Wert überwiegend um jeweils einen Prozentpunkt. In dem letzten Abschnitt mit den FS40-Prompts zeigt sich, dass sowohl die Persona als auch COT einen negativen Einfluss auf den F1-Score haben. In Kombination können sie, mit Ausnahme der Behauptungen, den Wert halten, benötigen dafür jedoch mehr Tokens.

4 Diskussion und Handlungsempfehlungen

Aus den Ergebnissen geht hervor, dass die reine Anzahl der übergebenen Einzelbeispiele für die Argumentationskomponenten und Beziehungen nicht auf die Genauigkeit der Identifikation zurückschließen lässt. So erzielten die Hauptaussagen für alle Prompts, in denen mindestens ein Beispiel enthalten ist, den höchsten F1-Score, obwohl sie in niedrigster Anzahl in einem Aufsatz enthalten sind. Vermutlich ist das Muster leichter für das LLM identifizierbar. Zudem wurde die Abhängigkeit der aufeinander aufbauenden Teilaufgaben deutlich. Es ist notwendig, dass die argumentativen Textabschnitte korrekt identifiziert und als korrekte Argumentationskomponente klassifiziert werden. Sofern dann nur ein Bestandteil des Beziehungstupels nicht korrekt ist, gilt die Beziehung als falsch. Zusätzlich ist zu bedenken, dass der BLEU-Score als verwendete Ähnlichkeitsmetrik zum Vergleich der Texte keine semantische Ähnlichkeit berücksichtigt. Somit werden Aussagen, die sinngemäß vom LLM korrekt wiedergegeben wurden, aber nicht ausreichend mit der exakten Formulierung des Aufsatzes übereinstimmen, als falsch gewertet. Hinzu kommt die Eigenheit des Datensatzes, dass bei mehreren Hauptaussagen aus den Annotationen der Beziehungen nicht hervorgeht, auf welche Hauptaussage sie sich bezieht. Dieser komplexe Zusammenhang zeigt sich in dem niedrigen F1-Score für die Beziehungen im Vergleich zu den Argumentationskomponenten.

Im Hinblick auf die Anzahl der im Prompt übergebenen Beispiele wurde deutlich, dass die Ergänzung von fortlaufend mehr Beispielen nicht zwangsläufig zu besseren Ergebnissen führt. Der größte Leistungszuwachs erfolgt bei der Übergabe des ersten Beispiels. Dies erscheint nachvollziehbar, da das LLM anhand des Beispiels relevante Merkmale identifizieren kann. Jedes weitere Beispiel trägt zwar zur Verfeinerung der Modellantworten bei, zeigt jedoch eine abnehmende Wirkung auf die Leistungssteigerung. Dies deutet darauf hin, dass das LLM bereits mit einer geringen Anzahl an Beispielen wesentliche Muster erkennen kann und ab 10 Beispielen die zusätzlichen Beispiele nur noch marginal zur Verbesserung beitragen. Es verhält sich somit vergleichbar zum Gesetz des abnehmenden Grenznutzens (Gossen, 1854, S. 4–5). Anhand der FS40-Prompts wurde sogar deutlich, dass sich Leistung auch verschlechtern kann. Da die Prompts vom Umfang so aufgebaut wurden, dass sie innerhalb des Kontextfensters des LLMs liegen und somit der Informationsverlust des LLMs ausgeschlossen werden kann, könnte dieser Leistungsabfall ein

Anzeichen für Overfitting sein (Google, 2024). Eine weitere Erhöhung der Beispiele für nachfolgende Untersuchungen wird deshalb als nicht sinnvoll erachtet.

Die Einbindung von COT erzielt im Vergleich zur Aufgabenbeschreibung in den meisten Fällen keine Leistungsverbesserung. Dies legt nahe, dass die explizite Vorgabe von Zwischenschritten zur Erreichung des gewünschten Ergebnisses nicht zwangsläufig einen Mehrwert für das Argument Mining bietet. Es ist jedoch auch denkbar, dass die Verwendung der strukturierten Ausgaben des LLMs die Wirkung von COT verringert, indem die vorformatierte Antwortstruktur möglicherweise die schrittweise Herleitung der Ergebnisse unterdrückt. Zukünftige Untersuchungen könnten diese Vermutung analysieren. Dem Gegenüber führt die Ergänzung einer Persona zu einer Erhöhung des F1-Scores. Dies scheint ein Verhaltensmuster vorzugeben, welches für das Argument Mining förderlich ist. In einigen Fällen erreicht die Kombination beider Ansätze einen höheren F1-Score als einzeln. Dies deutet darauf hin, dass sich die beiden Techniken nicht grundsätzlich gegenseitig ausschließen, sondern unter bestimmten Bedingungen einen Synergieeffekt erzielen können.

Entlang der Untersuchung wurde deutlich, dass es sich beim Argument Mining um ein komplexes Forschungsgebiet handelt, in welchem ein Mangel an hochwertig annotierten Datensätzen und einheitlichen Definitionen vorliegt. Die Ergebnisse verdeutlichen, dass der Einsatz von Prompt Engineering Techniken einen signifikanten Einfluss auf die Leistungsfähigkeit von LLMs im Argument Mining hat. So konnte ein maximaler Leistungszuwachs von 12 % für die Beziehungen bis hin zu 41 % für die Hauptaussagen erzielt werden. Es konnte jedoch auch gezeigt werden, dass die Ergänzung von COT in einem ZS-Prompt zu einer Verschlechterung der Leistung von bis zu 5 % führen kann. Ein gleichbleibender F1-Score bei einer zunehmenden Tokenanzahl kann als eine Verschlechterung verstanden werden, da sich damit auch die Kosten erhöhen. In solch einer Situation sollte der Prompt mit weniger Tokens verwendet werden. Der F1-Score pro eingesetzten Tokens ist für die ZS-Prompts zwar am höchsten, jedoch ist der F1-Score so gering, dass sie nicht bevorzugt werden sollten. Dieses Verhältnis sinkt für die restlichen Prompts stark ab, da ein einzelnes Input-Output-Paar fast 16-mal so viele Tokens umfasst wie die Aufgabenbeschreibung. Im Hinblick auf die bisherigen Ergebnisse wird deutlich, dass es keine eindeutige Prompt Engineering Technik gibt, die über alle

Argumentationskomponenten sowie die Beziehungen durchgehend den höchsten F1-Score erzielt. Jedoch ist die Anzahl der übergebenen Beispiele maßgeblich für die Maximierung des F1-Scores. Es konnte gezeigt werden, dass sich die leistungssteigernden Effekte mit der Ergänzung einer Persona mit bzw. ohne COT mit zunehmender Anzahl an im Prompt enthaltenen Beispielen verringern. Dies deutet darauf hin, dass die Leistungssteigerung durch diese Ergänzungen insbesondere dann relevant ist, wenn die Anzahl der Beispiele gering ist. Der effiziente Umgang mit der Tokenanzahl ist ein wichtiger Aspekt bei der Anwendung von LLMs. Je mehr Beispiele ein Prompt umfasst, desto größer wird die Tokenanzahl und umso höher sind auch die Kosten des Prompts. Welcher Prompt als geeignet gilt, hängt somit von den spezifischen Anforderungen des Anwendungsfalls, wie der Maximierung des F1-Scores oder der Einhaltung einer Kostenobergrenze, ab. Generell erscheint es sinnvoll, einen ausgeglichenen Ansatz zu verfolgen, bei dem ein Prompt weniger Beispiele und dafür die Ergänzung einer Persona und je nach Argumentationskomponente in Verbindung mit COT enthält.

Die vorgestellten Ansätze erreichen einen niedrigeren F1-Score als die von Stab und Gurevych (2017b, S. 646) oder Cabessa et al. (2024, S. 5) durchgeführten Untersuchungen für den gleichen Datensatz. Auch wenn die Einschätzung, ab wann die Leistung des Modells ausreichend ist, individuell vom konkreten Anwendungsfall abhängt, erscheinen die mit der Anwendung von Prompt Engineering Techniken erzielten Leistungen als noch nicht ausreichend, um als verlässliche Alternative zu den bisherigen Ansätzen zu gelten. Der Fokus dieser Untersuchung lag jedoch auch auf der Analyse der Effekte einzelner Prompt Engineering Techniken für das Argument Mining. Hierfür konnte gezeigt werden, dass sich die Generalisierungsfähigkeiten des LLMs auch für solch komplexe Anwendungsfälle mittels Prompt Engineering gezielt steuern lassen. Der Vorteil von Prompt Engineering liegt darin, dass es wesentlich leichter umzusetzen ist im Vergleich zum Fine-Tuning oder Pre-Training und sich besonders für Anwendungsfälle ohne ausreichende Menge an annotierten Daten eignet.

Es ist zu bedenken, dass die Ergebnisse nur für das untersuchte LLM GPT-4o mini gelten. Sowohl die Verwendung von abweichenden Formulierungen in den Prompts als auch die Reihenfolge des Inhalts können zu unterschiedlichen Antworten des LLMs führen (Google, 2024). Die Ergebnisse beziehen sich folglich auf die

dargestellten Prompt Templates. Der Datensatz enthält lediglich argumentative Aufsätze in englischer Sprache, weshalb die Prompts auch nur Beispiele von argumentativen Aufsätzen beinhalten. Die Prompts sind somit vermutlich nicht domänenunabhängig anwendbar. Trotz der zuvor beschriebenen Einschränkungen gibt die vorliegende Untersuchung Aufschluss darüber, wie leistungsfähig LLMs für das Argument Mining bei überzeugenden Aufsätzen 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. Auf Grundlage dieser Ergebnisse können weitere Untersuchungen anschließen. Im Sinne der Optimierung des LLMs und unter der Voraussetzung, dass ein ausreichend großer Datensatz vorliegt, könnte ein Fine-Tuning des LLMs 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. Auch die Verbesserung der Aufgabenbeschreibung beispielsweise anhand von strukturellen sowie kontextuellen Informationen oder der dynamischen Auswahl der Beispiele in Abhängigkeit des Eingabetextes, wie in Cabessa et al. (2024) durchgeführt wurde, könnte dazu beitragen, die Leistung des LLMs weiter zu steigern. Sollten diese Ansätze nicht möglich sein, ist die Untersuchung weiterer, leistungsstärkerer LLMs ebenfalls ein vielversprechender Ansatz. Diese sind zum aktuellen Zeitpunkt vermutlich jedoch kostenintensiver. Sofern ein LLM dann eine ausreichende Performance erzielt, wäre es interessant, darauf aufbauend den Ansatz so weiterzuentwickeln, dass die Argumentationsstrukturen in Strukturdiagramme überführt werden, um die extrahierte Argumentation visuell leicht verständlich aufzuarbeiten. Auch die Betrachtung weiterer Sprachen oder anderer argumentativer Texte aus unterschiedlichen Domänen ist denkbar.

5 Quellenverzeichnis

Anadkat, 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.

Cabessa, J., Hernault, H., & Mushtaq, U. (2024). *In-Context Learning and Fine-Tuning GPT for Argument Mining* (arXiv:2406.06699). arXiv. https://doi.org/10.48550/arXiv.2406.06699

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. (o. J.). *Prompting*. How-to Guides. Abgerufen 15. Dezember 2024, von https://www.llama.com/docs/how-to-guides/prompting/

OpenAI. (o. J.-a). *Advanced usage*. OpenAI Platform. Abgerufen 3. Januar 2025, von https://platform.openai.com

OpenAI. (o. J.-b). *Batch API*. OpenAI Platform. Abgerufen 17. Januar 2025, von https://platform.openai.com/docs/guides/batch/batch-api

OpenAI. (o. J.-c). *Chat. Temperature*. OpenAI Platform. API Reference. Abgerufen 20. Dezember 2024, von https://platform.openai.com/docs/api-reference/chat/create

OpenAI. (o. J.-d). *Models*. OpenAI Platform. Abgerufen 22. Dezember 2024, von https://platform.openai.com/docs/models/

OpenAI. (o. J.-e). *Optimizing LLM Accuracy*. OpenAI Platform. Docs. Abgerufen 20. Dezember 2024, von https://platform.openai.com/docs/guides/optimizing-llm-accuracy

OpenAI. (o. J.-f). *Pricing*. Abgerufen 17. Januar 2025, von https://openai.com/api/pricing/

OpenAI. (o. J.-g). *Prompt engineering*. OpenAI Platform. Docs. Abgerufen 20. Dezember 2024, von https://platform.openai.com/docs/guides/prompt-engineering

OpenAI. (o. J.-h). *Rate limits*. OpenAI Platform. Abgerufen 17. Januar 2025, von https://platform.openai.com/docs/guides/rate-limits?context=tier-free

OpenAI. (o. J.-i). *Structured Outputs*. Docs. Abgerufen 24. Dezember 2024, von https://platform.openai.com/docs/guides/structured-outputs

OpenAI. (o. J.-j). *Text generation*. Abgerufen 18. Januar 2025, von https://platform.openai.com/docs/guides/text-generation

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

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

6 Anhang

6.1 Prompt-Bausteine

Die nachfolgenden Prompt-Bausteine dienen zum Verständnis der angewendeten Prompts. Die 20 Prompt-Variationen können in dem Repository eingesehen werden.

Aufgabenbeschreibung

You will be given a text. Extract the argumentative units major claim, claim, and premise as parts of the text. Also extract the argumentative relationships between the units. Claims can be for or against the major claims. Premises, on the other hand, can support or attack a claim or another premise. There may be several major claims. Return the argumentative units and the relationships between them as a JSON object.

Input-Output-Paar

An dieser Stelle ist schematisch der Aufbau eines Input-Output-Paars dargestellt.

Here are x examples of text and their corresponding json data: # Example x ## Input [text of essay] ## Output { "MajorClaims": [{"ID": "MC1", "Text": "Text"}, {"ID": "MC2", "Text": "Text"}, ...], "Claims": [

```
        {"ID": "C1", "Text": "Text"},
        {"ID": "C2", "Text": "Text"},
        ...
    ],
    "Premises": [
        {"ID": "P1", "Text": "Text"},
        {"ID": "P2", "Text": "Text"},
        ...
    ],
    "ArgumentativeRelations": [
        {"Origin": "C1", "Relation": "for", "Target": "MC"},
        {"Origin": "C2", "Relation": "against", "Target": "MC"},
        {"Origin": "P1", "Relation": "supports", "Target": "C1"},
        {"Origin": "P2", "Relation": "attacks", "Target": "C2"},
        ...
    ]
}
```

Persona

```
You are a expert in Argument Mining and therefore a master at the annotation of argumentative components and their relationships in a text.
```

Chain-of-Thought

```
# Instructions:

1. **Extract the argumentative text parts:**
```

Identify all the relevant parts of the text that contain arguments, including main ideas and supporting or attacking details.

2. **Identify the argumentative components:**

Label the extracted text parts as one of the following:

- Major claim (MC): The author´s standpoint on the topic.

- Claim (C): Statements that can be for or against a major claim.

- Premise (P): Evidence or reasoning that support or attacka a claim or another premise.

3. **Determine the relationships between components:**

- For claims, identify whether they are “for” or “against” the major claim.

- For premises, identify whether they “support” or “attack” a claim or another premise.

4. **Format the output:**

Present the extracted argumentative components and relationships in the following JSON format:

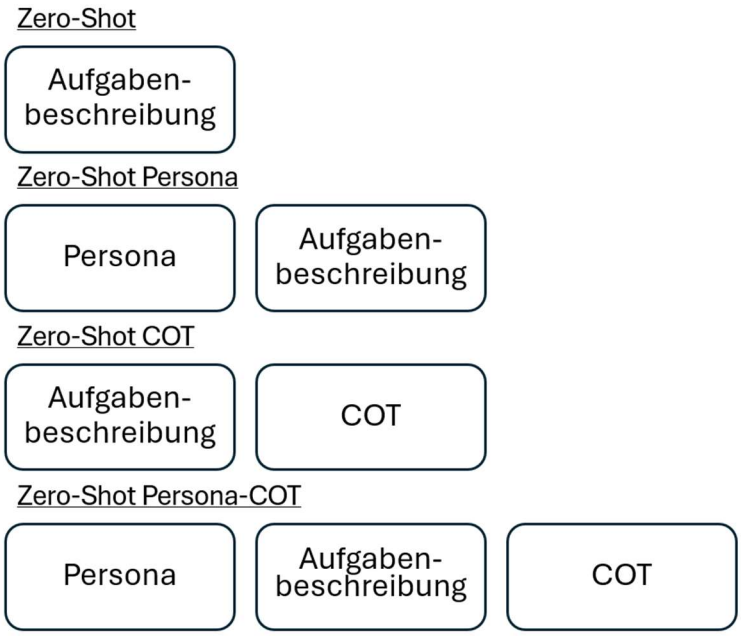
```json
{
   "MajorClaims": [
      {"ID": "MC1", "Text": "Text"},
      {"ID": "MC2", "Text": "Text"},
   ],
   "Claims": [
      {"ID": "C1", "Text": "Text"},
      {"ID": "C2", "Text": "Text"},
   ],
   "Premises": [
```

```
        {"ID": "P1", "Text": "Text"},
        {"ID": "P2", "Text": "Text"},
    ],
    "ArgumentativeRelations": [
        {"Origin": "C1", "Relation": "for", "Target": "MC"},
        {"Origin": "C2", "Relation": "against", "Target": "MC"},
        {"Origin": "P1", "Relation": "supports", "Target": "C1"},
        {"Origin": "P2", "Relation": "attacks", "Target": "C2"},
        ...
    ]
}
```

6.2 Prompt Templates

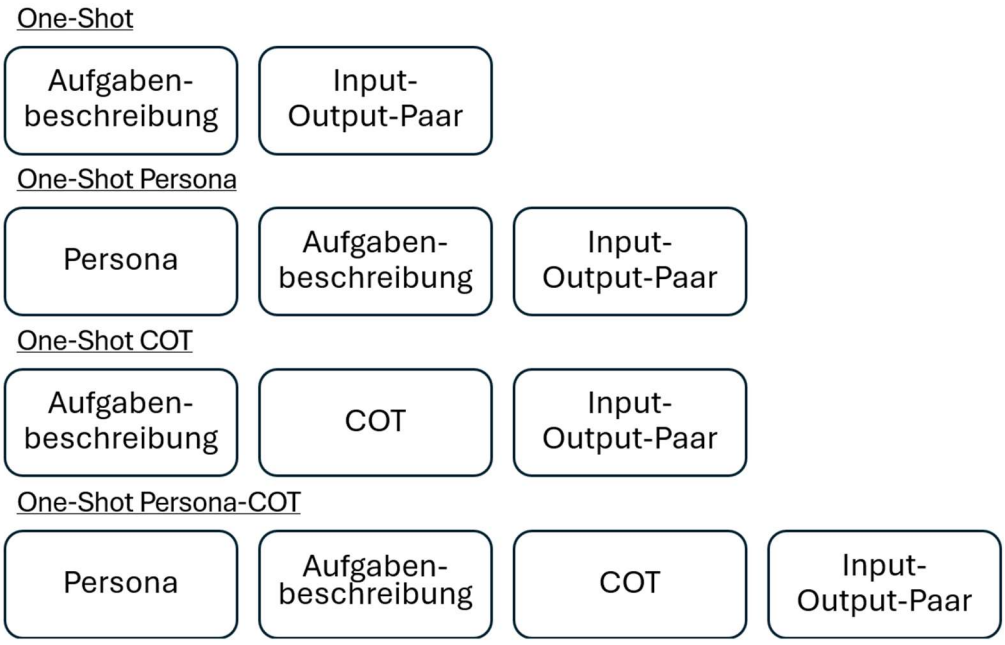
Die hier aufgeführten Abbildungen zeigen den modularen Aufbau der zuvor gezeigten Prompt-Bausteine. Die Few-Shot-Prompts sind systematisch gleich zu den One-Shot-Prompts aufgebaut, jedoch werden mehrere Input-Output-Paare als Beispiele übergeben.

Abbildung 4
Zero-Shot Prompt-Struktur



Eigene Darstellung.

Abbildung 5
One-Shot Prompt-Struktur



Eigene Darstellung.

6.3 Übersicht der Tokenanzahl pro Prompt

Die nachfolgende Tabelle enthält eine Übersicht über alle Prompt-Variationen und die dazugehörige Tokenanzahl. Diese steigt mit zunehmender Komplexität sichtlich an.

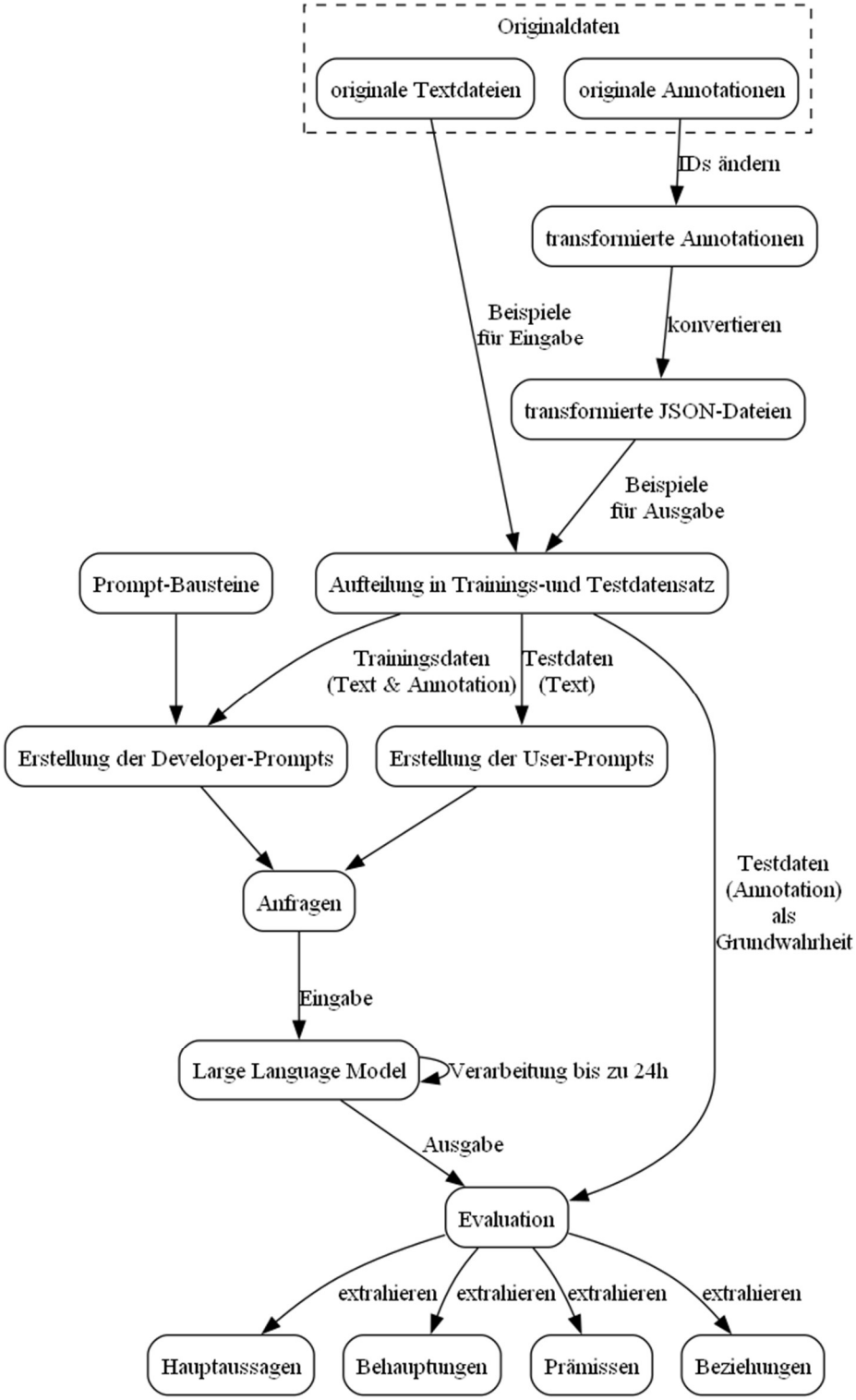
Tabelle 1
Tokenanzahl pro Prompt

Lfd.-Nr.	Prompt	Tokenanzahl
1	zero-shot	82
2	zero-shot-persona	105
3	zero-shot-cot	480
4	zero-shot-persona-cot	503
5	one-shot	1.780
6	one-shot-persona	1.790
7	one-shot-cot	2.166
8	one-shot-persona-cot	2.189
9	few-shot-10	13.848
10	few-shot-10-persona	13.871
11	few-shot-10-cot	14.247
12	few-shot-10-persona-cot	14.270
13	few-shot-20	27.681
14	few-shot-20-persona	27.704
15	few-shot-20-cot	28.080
16	few-shot-20-persona-cot	28.103
17	few-shot-40	54.048
18	few-shot-40-persona	54.071
19	few-shot-40-cot	54.447
20	few-shot-40-persona-cot	54.470

6.4 Prozess der Untersuchung

Die Abbildung 6 stellt den Prozess von der Datenaufbereitung bis hin zur Evaluation schematisch dar.

Abbildung 6
Prozessschema der Untersuchung



Eigene Darstellung.