Universität Leipzig
Medizinische Fakultät
Institut für Medizinische Informatik, Statistik und Epidemiologie

QUESTION ANSWERING AUF DEM LEHRBUCH „HEALTH INFORMATION SYSTEMS“ MIT HILFE VON UNÜBERWACHTEM TRAINING EINES PRETRAINED TRANSFORMERS

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science
(M. Sc.)

vorgelegt von

Paul Keller
Studiengang Medizininformatik M. Sc.

Leipzig, den 30.09.2023

AUTOR:
Paul Keller
Geboren am 23.05.1998 in Leipzig, Deutschland

TITEL:
Question Answering auf dem Lehrbuch „Health Information Systems“ mit Hilfe von unüberwachtem Training eines Pretrained Transformers

INSTITUT:
Institut für Medizinische Informatik, Statistik und Epidemiologie
Medizinische Fakultät Universität Leipzig

REFERENT:
Prof. Dr. Alfred Winter

BETREUER:
Dr. Konrad Höffner

ABSTRAKT

Die Extraktion von Wissen aus Büchern ist essentiell und komplex. Besonders in der Medizininformatik ist ein einfacher und vollständiger Zugang zu Wissen wichtig. In dieser Arbeit wurde ein vortrainiertes Sprachmodell verwendet, um den Inhalt des Buches *Health Information Systems* von Winter u. a. (2023) effizienter und einfacher zugänglich zu machen. Während des Trainings wurde die Qualität des Modells zu verschiedenen Zeitpunkten evaluiert. Dazu beantwortete das Modell Prüfungsfragen aus dem Buch und aus Modulen der Universität Leipzig, die inhaltlich auf dem Buch aufbauen. Abschließend wurde ein Vergleich zwischen den Trainingszeitpunkten, dem nicht weiter trainierten Modell und dem Stand der Technik Modell GPT4 durchgeführt. Mit einem MakroF1-Wert von 0,7 erreichte das Modell GPT4 die höchste Korrektheit bei der Beantwortung der Klausurfragen. Diese Leistung konnte von den anderen Modellen nicht erreicht werden. Allerdings stieg die Leistung von einem anfänglichen MakroF1-Wert von 0,13 durch kontinuierliches Training auf 0,33. Die Ergebnisse zeigen eine deutliche Leistungssteigerung durch diesen Ansatz und bieten eine Grundlage für zukünftige Erweiterungen. Damit ist die Machbarkeit der Beantwortung von Fragen zu Informationssystemen im Gesundheitswesen und der Lösung einer Beispielklausur mit Hilfe von weiter trainierten Sprachmodellen gezeigt, eine praktische Anwendung erreichen diese Modelle jedoch nicht, da sowohl die Leistung unter dem aktuellen Stand der Technik liegt als auch die hier vorgestellten Modelle einen Großteil der gestellten Fragen nicht vollständig korrekt beantworten können.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

AKRONYME

API	Application Programming Interface
BeLL	Besondere Lernleistung
BERT	Bidirectional Encoder Representations from Transformers
BPE	Byte Pair Encoding
CPU	Central Processing Unit
ELMo	Embeddings from Language Models
GMDS	Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie e.V.
GPT	General Pretrained Transformer
GPU	Graphics Processing Unit
LLaMA	Large Language Model Meta AI
LLM	Large Language Model
LoRA	Low-Rank Adapation
NLP	Natural Language Processing
NN	Neuronales Netzwerk
PaLM	Pathways Language Model
PIP	Package Installer for Python
QA	Question Answering
QAS	Question Answering System
RAM	Random Access Memory
SNIK	Semantisches Netz des integrierten Informationsmanagements im Krankenhaus
SOTA	State of the Art (Stand der Technik)
URL	Uniform Resource Locator
ZeRO	Zero Redundancy Optimizer

1 EINLEITUNG

1.1 GEGENSTAND

Eine effektive und effiziente Wissensbeschaffung bildet einen fundamentalen Bestandteil einer qualitativ hochwertigen klinischen Praxis in der Medizin. Bei jeder medizinischen Handlung werden große Mengen an Wissen genutzt und erzeugt, sei es als Grundlage für eine Diagnose oder zur Dokumentation des Behandlungsprozesses. Die strukturierte und klassifizierte Speicherung und Wiedergabe dieses Wissens ist ein kontinuierlicher Entwicklungsprozess und Gegenstand aktueller Forschung.

Die Digitalisierung der Medizin ist ein weites Themenfeld mit stetig wachsendem Bedarf. Die Medizinische Informatik beschreibt dabei „die Wissenschaft der systematischen Erschließung, Verwaltung, Aufbewahrung, Verarbeitung und Bereitstellung von Daten, Informationen und Wissen in der Medizin und im Gesundheitswesen. [...]“[1]. Vor diesem Hintergrund gewinnt die Entwicklung und Implementierung effizienter Informationssysteme und Technologien zur Unterstützung der klinischen Praxis zunehmend an Bedeutung.

In der Lehre wird die Praxis der Medizinischen Informatik durch umfangreiche Literatur, z.B. in Winter u. a. (2023), unterstützt. Zur Strukturierung von Fachbegriffen und Rollen des Informationsmanagements im Krankenhaus existiert die Ontologie Semantisches Netz des integrierten Informationsmanagements im Krankenhaus (SNIK) (Jahn u. a., 2014), folgend der Metaontologie SNIK und Teil des Projekts SNIK des Instituts für Medizinische Informatik, Statistik und Epidemiologie[2] an der Universität Leipzig. Die Nutzung dieses Netzes ermöglicht eine systematische Darstellung von Rollen, Entitäten und Funktionen des Informationsmanagements im Krankenhaus, unabhängig von der Definition der zugrunde liegenden Literaturquellen.

Die Bedeutung von maschinellem Lernen, Deep Learning und Sprachmodellen ist in der heutigen Zeit sehr präsent. Diese Technologien werden in vielen Bereichen, von der Automobilindustrie bis hin zu

1 Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie e.V. (GMDS) (2023). Definition Medizinische Informatik. https://www.gmds.de/aktivitaeten/medizinische-informatik/ (abgerufen am 7.3.2023).

2 Institut für Medizinische Informatik, Statistik und Epidemiologie. https://www.imise.uni-leipzig.de/Institut (besucht am 9.3.2023).

medizinischen Anwendungen, eingesetzt, um neue Methoden der Wissensgewinnung und -verarbeitung zu ermöglichen. Verschiedene KI-Modelle können in vielen Bereichen, wie z.B. der Erkennung seltener Krankheiten (Brasil u. a., 2019), der personalisierten Medizin (Johnson u. a., 2021) oder dem Marketing (Davenport u. a., 2020) weitreichende Auswirkungen auf zukünftige Arbeitsprozesse haben. Sprachmodelle wie das General Pretrained Transformer (GPT)-3-Modell von OpenAI (Brown u. a., 2020) können dazu beitragen, Texte in verschiedenen Sprachen automatisch zu übersetzen und sogar kreatives Schreiben zu ermöglichen. Deep Learning, das auf künstlichen neuronalen Netzen basiert, ermöglicht eine noch tiefere und komplexere Verarbeitung von Daten. In der Medizin kann Deep Learning beispielsweise zur Diagnose von Krankheiten und zur Analyse medizinischer Bilder eingesetzt werden (Esteva, Kuprel, Novoa u. a., 2017).

1.2 PROBLEMSTELLUNG

Informationssysteme im Gesundheitswesen sind komplex, abhängig von den Anforderungen der jeweiligen Organisation und unterliegen einer ständigen Weiterentwicklung. Die Anforderungen sind vielfältig und umfassen beispielsweise die Speicherung und Verarbeitung von Patientendaten, die Dokumentation von Behandlungsprozessen, die Unterstützung von Diagnose- und Therapieprozessen sowie die Unterstützung von Forschungsaktivitäten. Ausgehend von diesen Anforderungen gibt es eine Fülle von Literatur, die sich mit der Definition, Implementierung und Wartung von Informationssystemen im Gesundheitswesen befasst. Das Buch *Health Information Systems* von Winter u. a. beschäftigt sich umfassend mit diesen Anforderungen und ist im März 2023 in der 3. Auflage erschienen (Winter u. a., 2023).

Das Management von Informationssystemen ist eine anspruchsvolle Aufgabe, die sich nicht nur auf die Anwendung durch das Krankenhauspersonal beschränkt. Es ist auch von großer Bedeutung für Studierende, um ein besseres Verständnis der bestehenden Systeme zu erlangen, und für Forschende, um diese Systeme zu erweitern oder neue Managementmethoden zu entwerfen. Eine konkrete und konsistente Wissensgrundlage ist sowohl für die Anwendung als auch für Lehre und Forschung von großer Bedeutung.

Eine weitere Herausforderung bei der Auseinandersetzung mit der Literatur zu Informationssystemen ist die Komplexität der Übertragung von theoretischen Konzepten auf praktische Anwendungsfälle. Insbesondere für Studierende und Praktiker erfordert die praktische Anwendung ein tiefes Verständnis der Zusammenhänge und der Anwendbarkeit der vorgestellten Konzepte auf konkrete Arbeitsumgebungen. Da die verfügbare Literatur oft sehr umfangreich und ihre

Definitionen fragmentiert sind, stellt die Identifikation relevanten Wissens für spezifische Problemstellungen eine weitere Herausforderung dar. Die Fragmentierung von Definitionen bezieht sich hier auf die Erläuterung eines Fachbegriffs oder Konzepts innerhalb einer Literaturquelle. Diese Definitionen werden häufig nicht zusammenfassend aufgelistet, sondern ergeben sich im Laufe der Texte und Kapitel. Dies führt dazu, dass große Teile eines Buches gelesen werden müssen, um einzelne Konzepte und ihre Beziehungen zu anderen Themen vollständig zu erfassen.

Der Umfang der Literaturquellen und die Fragmentierung bei der Definition von Fachbegriffen erschweren eine schnelle Wissensbeschaffung, insbesondere für Studierende, die grundlegende Konzepte richtig verstehen wollen.

- Problem: Schwierigkeiten bei der Wissensbeschaffung aufgrund des Umfangs von Winter u. a. (2023) und der Fragmentierung von Definitionen

1.3 MOTIVATION

Eine Strukturierung von Wissen zum Management von Krankenhausinformationssystemen ist bereits Bestandteil des Projekts SNIK (Jahn u. a., 2014; Schaaf u. a., 2015), eine Ontologie des Instituts für Medizinische Informatik, Statistik und Epidemiologie der Universität Leipzig. Auf Basis dieser Ontologie wurden bereits verschiedene Methoden zur Wissensextraktion untersucht.

Die Besondere Lernleistung (BeLL) mit dem Titel „Question Answering on SNIK“ (Brunsch, 2022; Brunsch, 2018) erweiterte den Zugang zu diesem Netz durch die Verwendung natürlicher englischer Sprache. Die Ergebnisse der BeLL wurden in einem Question Answering System (QAS) QAnswer (QAnswer, 2023) umgesetzt, zeigen aber Defizite in der Erklärbarkeit und Verständlichkeit komplexerer Fragen. Im Gegensatz dazu untersuchte Omar u. a. (2023) auf einem anderen Datensatz die Leistung eines Sprachmodells (in diesem Fall ChatGPT) im Vergleich zu einer herkömmlichen, auf einem Wissensgraphen basierenden QAS (verwendet wurde $KGQA_N$). Die Ergebnisse zeigten, dass ChatGPT im Vergleich zum verwendeten $KGQA_N$ erstaunlich stabile und verständliche Antworten lieferte, jedoch bei der Ausgabe korrekter Antworten deutlich schlechter abschnitt. Hier steht zu erwarten, dass durch ein besseres Training die Anzahl der falschen Antworten deutlich reduziert werden kann.

Es ist daher notwendig zu untersuchen, ob der Einsatz eines Sprachmodells die Anwendungsschwierigkeiten in QAnswer beheben kann

und ob die Wiedergabe mit niedrigem Wahrheitswert durch zusätzliches Training verbessert werden kann.

1.4 ZIELSETZUNG

Dem in Abschnitt 1.2 gezeigten Problem werden folgende Ziele dieser Arbeit zugeordnet.

- Ziel Z1: Beantwortung von Fragen zu Informationssystemen im Gesundheitswesen in natürlicher Sprache durch ein Sprachmodell mit Hilfe von Winter u. a. (2023)
- Ziel Z2: Lösung einer Beispielklausur des Moduls „Architektur von Informationssystemen im Gesundheitswesen“[3] mit Hilfe eines Sprachmodells. Das Ziel ist kein produktives System, sondern soll lediglich die Machbarkeit der Beantwortung von Fragen mit Hilfe eines Sprachmodells aufzeigen.

1.5 BEZUG ZU ETHISCHEN LEITLINIEN DER GMDS

Die ethischen Leitlinien der GMDS (Ahrens u. a., 2023) geben „sowohl den tragenden Gesellschaften als Institution als auch dem einzelnen Mitglied eine Orientierung, welche ethischen Forderungen in ihrem bzw. seinem jeweiligen Aufgaben- und Verantwortungsbereich relevant sein können“ (Ahrens u. a., 2023). Aufgeteilt in 16 Artikel werden hier verschiedene Kompetenzen und Verantwortlichkeiten definiert, unter die auch das hier beschriebene System fällt. Da das zu entwickelnde Sprachmodell Wissen über medizinisches Wissen, insbesondere über Informationssysteme im Gesundheitswesen, bereitstellt, unterliegt es in seiner Existenz als „Wissensbasis“ einer Vielzahl von Artikeln. Es wirkt unterstützend in den Artikeln 1 „Auftrag“, 2 „Fachkompetenz“, 3 „Kommunikative Kompetenz“, kann aber durch unsachgemäßen Gebrauch und seines zugrundeliegenden Wesens entgegen den Artikeln 1 „Auftrag“, 2 „Fachkompetenz“, 3 „Kommunikative Kompetenz“, 4 „Medizinethische Kompetenz“, 6 „Soziale Verantwortung“ und 13 „Forschung“ handeln.

In Artikel 1 „Auftrag“ heißt es unter anderem, dass „Die Würde des Menschen und das Persönlichkeitsrecht [...] dabei vorrangig geachtet und geschützt werden [müssen]“ (Ahrens u. a., 2023). Dazu gehört insbesondere die „Allgemeine Erklärung der Menschenrechte“, in der das „Verbot der Diskriminierung z.B. nach Geschlecht oder [aus rassistischen Gründen] (Art. 2, 7)“ verankert ist und der „Schutz des (geistigen) Eigentums (Art. 17, 27)“ und die „Gedanken-, Gewissens-, Religions- und Meinungsfreiheit (Art. 18, 19)“ beschrieben werden.

3 einem Modul des Masterstudiengangs Medizinische Informatik an der Universität Leipzig, das inhaltlich auf Winter u. a. (2023) aufbaut

Die von dem Sprachmodell zu generierenden Antworten ergeben sich aus einer Vielzahl von zuvor genutzten Datensätzen aus dem Internet, die nicht notwendigerweise diesen ethischen Leitlinien folgen. Eine inhaltliche Garantie für die Einhaltung dieser Leitlinien ist daher ohne zusätzliche Filterung der Antworten zu gestellten Fragen nicht möglich. Durch eine optimale Filterung können Antworten, die den Leitlinien widersprechen, ausgeschlossen werden. Diese Filterung ist jedoch aufgrund ihrer Komplexität sowohl in zeitlicher als auch in finanzieller Hinsicht nicht Bestandteil dieser Arbeit. Es muss daher davon ausgegangen werden, dass es durchaus Antworten geben kann, die gegen die Leitlinien verstoßen.

Artikel 1 wird durch das Sprachmodell gefördert, in dem es Medizinische Versorgungseinrichtungen durch effiziente und schnelle Wissensbeschaffung in ihren Fähigkeiten unterstützt, „ihre Leistungen qualitativ und quantitativ nachweisen, überwachen und sicherstellen [zu] können“ (Ahrens u. a., 2023).

Artikel 2 „Fachkompetenz“ definiert, dass das Mitglied seine „Fachkompetenz nach dem Stand der Wissenschaft und Technik erwirbt [...]“ und „Maßnahmen zur Fehlervermeidung ergreift“ (Ahrens u. a., 2023). Dies ist teilweise durch das Sprachmodell gegeben, da das extrahierte Wissen auf dem Stand des zugrundeliegenden Buches Winter u. a. (2023) aufbaut. Eine Fehlervermeidung ist hier jedoch nicht vorgesehen. Wie in Omar u. a. (2023) gezeigt, weist ChatGPT als GPT-Modell eine geringe Wahrheitsquote auf. Da es sich bei dem zu entwickelnden Sprachmodell um ein ähnliches Modell handelt, wird die fehlerfreie Beantwortung von Fragen zwar angestrebt, kann aber nicht garantiert werden.

Artikel 3 „Kommunikative Kompetenz“ beschreibt die Fähigkeit „Recht, Interessen [und] Konventionen der verschiedenen von [dem Mitglied] seiner Arbeit Betroffenen zu verstehen und zu berücksichtigen [...]“ und „Wissenschaftliche Erkenntnisse in verständlicher Form der Öffentlichkeit zugänglich [...]“ (Ahrens u. a., 2023) zu machen. Das Sprachmodell unterstützt dabei Fragende durch eine Antwort in möglicherweise verständlicherer Form der wissenschaftlichen Erkenntnisse in Winter u. a. (2023), jedoch ist hier nicht gegeben, dass die Betroffenen in der Antwort berücksichtigt oder verstanden wurden.

Artikel 4 „Medizinethische Kompetenz“ legt fest, dass das Mitglied „ethische Prinzipien der Medizin [...] bei seinem beruflichen Handeln beachtet“ (Ahrens u. a., 2023). Zu den ethischen Prinzipien (Ahrens u. a., 2023) gehören auch die „Achtung vor dem Menschen zu wahren“ und die „Würde des Individuums zu schützen“. Beides kann nicht

allein durch das Sprachmodell gewährleistet werden und muss durch mögliche Filter ergänzt werden.

Artikel 6 „Soziale Verantwortung“ definiert die Verantwortungen des Mitglied „gesellschaftliche Auswirkungen [zu] berücksichtigen [...]“ und die „Allgemeine Erklärung der Menschenrechte und ethische[n] Prinzipien der Medizin“ (Ahrens u. a., 2023) zu beachten. Wie bereits zu den Artikeln 1 und 4 ausgeführt, ist es dem Sprachmodell aufgrund der Datengrundlage nicht möglich, diesen Artikel, insbesondere die hier genannten Punkte, in seiner Antwort zu berücksichtigen.

Artikel 13 „Forschung“ definiert, dass das Mitglied „gute Wissenschaftliche Arbeit, insbesondere Offenheit und Transparenz [und] Akzeptanz von Kritik“ (Ahrens u. a., 2023) einhalten soll. Gute wissenschaftliche Arbeit wird weiter definiert als „die strikte Ehrlichkeit im Hinblick auf die Beiträge von Partnern, Konkurenten, Vorgängern zu wahren“ und „weder Fälschung oder Plagiate [zu] benutzen“ (Ahrens u. a., 2023). Da das Sprachmodell auf dem Inhalt eines Buches trainiert wird, in seiner Datenbasis aber bereits eine Vielzahl von Texten, darunter auch Plagiate und Fälschungen, gelernt hat, ist es nicht möglich, die Antworten als wissenschaftliche Quelle zu verwenden, da sowohl Plagiate als auch Fälschungen ohne Annotation vorhanden sein können. Die Antworten sollten als reine Wissensextraktion und nicht als Quelle für wissenschaftliche Arbeiten betrachtet werden.

1.6 AUFGABENSTELLUNG

Die in Abschnitt 1.4 genannten Ziele Z_i werden durch die hier aufgeführten Aufgaben A_i gelöst.

- Aufgabe zu Ziel Z1
  - Aufgabe A1.1: Es sollen aktuelle Sprachmodelle verglichen werden. Dabei sind die Einschränkungen der Verfügbarkeit und Verwendbarkeit zu berücksichtigen. GPT-Modelle basieren auf großen Datenmengen und enthalten mehrere Milliarden Parameter. Ein eigenes Training eines Modells würde sowohl den zeitlichen als auch den finanziellen Rahmen übersteigen, weshalb auf ein vortrainiertes Modell zurückgegriffen werden muss. Dazu muss das Modell frei verfügbar sein und unter einer Open-Source-Lizenz stehen. Außerdem muss das Modell am Rechenzentrum der Universität Leipzig geladen und trainiert werden können. Aufgrund der großen Anzahl an Parametern sind auch hier Grenzen des Arbeitsspeichers und damit der Größe des Modells gesetzt.

  - Aufgabe A1.2: Für ein effizientes und erfolgreiches Training des Sprachmodells ist eine Datenkuration von Winter u. a. (2023) notwendig. Abschnitte wie das Literaturverzeichnis, Aufzählungen oder Grafiken und deren Beschriftung müssen vor der Verwendung des Textes entfernt oder umgeschrieben werden.
  - Aufgabe A1.3: Das trainierte Sprachmodell wird zur Beantwortung von Fragen zu Winter u. a. (2023) verwendet. Dabei wird während des Trainings der in Brown u. a. (2020) beschriebene „Zero-Shot"-Ansatz verfolgt. Dies bedeutet, dass das Verständnis der Fragestellung und die Bewertung wichtigen Wissens allein durch das Modell erfolgt und nicht durch vordefinierte Fragen im Datensatz dem Modell beigebracht wird.

- Aufgabe zu Ziel Z2
  - Aufgabe A2.1: Das in dieser Arbeit gewählte Sprachmodell wird vor und nach dem Training hinsichtlich seiner Fähigkeit, Fragen korrekt zu beantworten, evaluiert. Dies ermöglicht eine Aussage über die Effektivität des Trainings und die Leistungssteigerung des Sprachmodells. Ebenso wird ein Vergleich mit dem aktuellen GPT-4 Modell (OpenAI, 2023) durchgeführt, um die Notwendigkeit eines Trainings zu ermitteln.
  - Aufgabe A2.2: Bewertung der Antwortoptionen von Klausurfragen nach gleichen Kriterien wie in Omar u. a. (2023)

1.7 AUFBAU DER ARBEIT

Kapitel 1 beschreibt das grundlegende Umfeld dieser Arbeit, formuliert existierende Probleme und Anforderungen, bietet Ziele zur Lösung dieser Probleme an und gibt Aufgaben, zur Umsetzung dieser Ziele. In Kapitel 2 werden Grundlagen gelegt zum Verständnis der in dieser Arbeit verwendeten Technologie, während in Kapitel 3 der aktuelle Stand der Forschung zusammengefasst wird. Kapitel 4 umfasst Lösungsstrategien des in Abschnitt 1.2 formulierten Problem mit einer anschließenden Beschreibung der Umsetzung dieser Lösungen in Kapitel 5. Die Ergebnisse dieser Arbeit werden in Kapitel 6 präsentiert und in Kapitel 7 zusammengefasst diskutiert. Kapitel 8 gibt abschließend einen Ausblick dieser Arbeit.

2 GRUNDLAGEN

Zum Verständnis der in Abschnitt 1.4 genannten Ziele und Ausführungen dieser Arbeit werden in diesem Kapitel Grundlagen zu den verwendeten Begriffen und Konzepten geschaffen. Das Kapitel gliedert sich in eine allgemeine Definition von Sprachmodellen in Abschnitt 2.1 mit einer genaueren Erläuterung der Architektur der Transformer. Danach folgt eine kurze Einführung in das Thema Neuronale Netze, in der die Funktionsweise von Neuronalen Netzen näher erläutert wird. Abschließend werden einige Begriffe aus der Medizinischen Informatik und der Datenverarbeitung in Abschnitt 2.3 erläutert.

2.1 SPRACHMODELLE

Definition 1 ***Sprachmodelle*** *sind neuronale Netze, welche menschliche Sprache verstehen und darauf reagieren.*

Sprachmodelle basieren auf künstlicher Intelligenz und werden in der Regel durch maschinelles Lernen trainiert. Sie werden in einer Vielzahl von Anwendungen eingesetzt, wie z.B. in der Übersetzung, Textgenerierung, Spracherkennung und Textklassifikation.

Definition 2 ***Autoregressive Sprachmodelle*** *sind Sprachmodelle, die Texte generieren, indem sie die Wahrscheinlichkeit von Wörtern oder Zeichen in einer Sequenz vorhersagen.*

Autoregressive Modelle basieren auf der Annahme, dass jedes Element in einer Sequenz von den vorhergehenden Elementen abhängt. In einem autoregressiven Sprachmodell wird die generative Wahrscheinlichkeit des nächsten Elements in der Sequenz geschätzt. Die Ausgabe im Kontext von Decoder-basierten Modellen ist daher eine Wahrscheinlichkeitsverteilung über das mögliche nächste Element. Das wahrscheinlichste Element wird dann der Sequenz hinzugefügt, und der Prozess wird wiederholt, bis die Sequenz vollständig ist. Üblicherweise werden autoregressive Sprachmodelle auf großen Textdatenmengen trainiert, um die statistischen Zusammenhänge in den Trainingsdaten zu erfassen.

2.1.1 *Transformer-Modelle*

Definition 3 *Ein* ***Transformer-Modell*** *ist ein Sprachmodell, das auf der Transformer-Architektur von Vaswani u. a. (2017) basiert.*

Im Gegensatz zu früheren Sprachmodellen, die auf rekurrenten oder faltenden neuronalen Netzen basieren, verwendet der Transformer einen Aufmerksamkeitsmechanismus, um die Beziehungen zwischen Wörtern in einem Satz zu modellieren. Transformer-Modelle bestehen aus einer Stapelung von Encoder- und Decoder-Schichten, wobei der Encoder dazu dient, Kontextinformationen aus der Eingabe zu extrahieren, während der Decoder die Ausgabe des Encoders verwendet, um die gewünschte Aufgabe zu lösen. Die Architektur eines Transformers ist in Abb. 2.1 dargestellt.

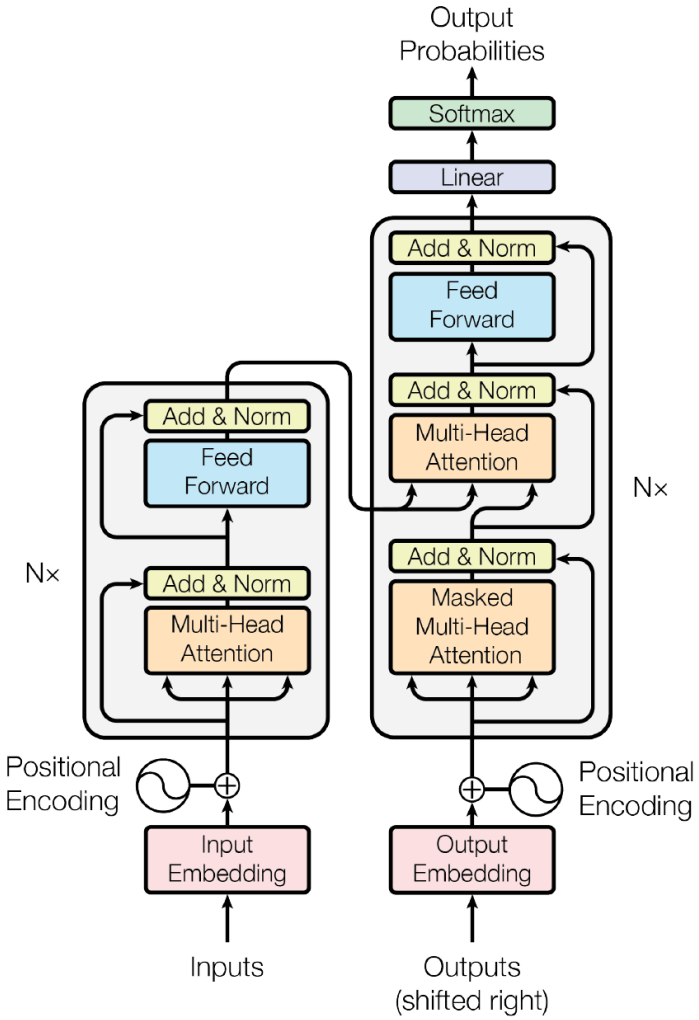

Abbildung 2.1: Die Transformer-Modell Architektur (Vaswani u. a., 2017)

Die erste schriftliche Erwähnung des Transformer-Modells sowie die Einführung der beiden Teilmodelle Encoder und Decoder findet sich in Vaswani u. a. (2017). Die hier beschriebene bidirektionale Architektur bildet die Grundlage für alle darauf aufbauenden Modelle und Weiterentwicklungen. Die Basisarchitektur wurde für verschiedene Anwendungen stark modifiziert. Seit 2017 gibt es grundlegende

Unterschiede in den Modellen und deren Möglichkeiten. Aus diesem Grund haben Kalyan, Rajasekharan und Sangeetha (2022) eine Taxonomie der Transformer-basierten vortrainierten Sprachmodelle eingeführt. Diese Taxonomie wird hier zur Beschreibung weiterer Architekturen und Methodiken verwendet.

Neben dem Grundbaustein eines Transformers — dem Aufmerksamkeitsmechanismus — sind zwei wichtige Modifikationen gegenüber normalen neuronalen Netzen in Transformer eingeflossen: Residuale Verbindungen und Dropout.

2.1.1.1 *Residuale Verbindungen*

Definition 4 ***Residuale Verbindungen*** *oder auch Deep Residual Connections, sind direkte Verbindungen zwischen Eingabe und Ausgabe und überspringen somit eine oder mehrere Schichten eines Neuronales Netzwerk (NN).*

Residuale Verbindungen als Level-Normalisierung ändern das Ziel eines NNs, behalten aber die gleichen Ausgaben durch ihre Level-Normalisierung bei. Dieses Konzept wurde erstmals in He u. a. (2016) eingeführt und bietet eine Lösung für ein grundlegendes Problem von großen, mehrschichtigen Transformer-Modellen. Bereits 2016 wurde im Bereich der Bilderkennung festgestellt, dass sich die Korrektheit von Modellen mit zunehmender Tiefe sättigt und sich dann rapide verschlechtert, wenn dieses Modell weiter trainiert wird. Dies setzte eine praktische Grenze für die Tiefe von NNs und verhinderte somit die Lösung komplexerer Probleme mit größeren Modellen. He u. a. (2016) beschreiben eine Lösung durch die genannten Residuen, die normale NN einfach ersetzen können, und zeigen auch die Effizienz dieser Methode.

2.1.1.2 *Dropout*

Die Einführung von Dropouts ist die zweite wichtige Änderung zur Weiterentwicklung der Transformer-Architektur.

Definition 5 ***Dropout*** *ist eine Methode, die die Trainingszeit von NNs verkürzt und die Generalisierung verbessert. Srivastava u. a. (2014) beschreiben die Methode als das zufällige Aussetzen von Neuronen unabhängig von der Eingabe in einem NN während des Trainings.*

Durch das Aussetzen von Neuronen wird das NN gezwungen, sich nicht auf andere Neuronen zu verlassen und somit eine bessere Generalisierung zu erreichen. Das Aussetzen erfolgt nur während des Trainings und nicht während der Inferenz. Die Methode wurde 2014 eingeführt und ist seitdem ein fester Bestandteil von NNs.

2.1.1.3 *Modellarten von Transformer-Modellen*

Basierend auf der eingeführten Transformer-Architektur wurden verschiedene Modellarten entwickelt, die nur Teile der Architektur nutzen und sich in ihrer Funktionsweise unterscheiden.

Definition 6 ***Encoder-basierte Modelle*** *sind Sprachmodelle, die hauptsächlich auf der Encoder-Architektur basieren. Sie wurden entwickelt, um Eingabesequenzen zu verarbeiten und eine kompakte Repräsentation (auch Kontextvektor genannt) zu erzeugen.*

Die zentrale Komponente eines Encoder-basierten Modells ist der Encoder. Er empfängt die Eingabesequenz und lernt schrittweise, die strukturellen und semantischen Informationen zu erfassen. Die so erzeugte Repräsentation des Encoders kann von anderen Modulen oder Schichten des Modells verwendet werden, um verschiedene Aufgaben wie z.B. die Klassifikation zu lösen. Encoder-basierte Modelle haben in der Regel keine Decoder-Schicht.

Definition 7 ***Decoder-basierte Modelle*** *sind Sprachmodelle, die auf einer speziellen Architektur basieren, bei der der Decoder eine zentrale Rolle spielt. Der Decoder ist eine Komponente, die darauf spezialisiert ist, aus einer gegebenen Eingabe eine sinnvolle und kohärente Ausgabe zu erzeugen.*

Im Zusammenhang mit der Textgenerierung erhält der Decoder normalerweise eine Sequenz von Vektoren als Eingabe, die von einem Encoder-Modul erzeugt wurde. Bei der Textgenerierung kann der Decoder aus einer gegebenen Ausgangssequenz fortlaufend neue Wörter oder Sätze generieren, um einen zusammenhängenden Text zu erzeugen.

2.1.2 *Transformer-spezifische Architekturen*

Definition 8 *Der* ***Self-Attention-Mechanismus*** *ist ein zentrales Konzept in der Transformer-Architektur und beschreibt die Gewichtung der Eingabesequenz auf Basis eines aktuellen Tokens.*

Hier wird ein Eingabevektor in drei separate Vektoren transformiert: Den Abfragevektor (engl. Query), den Schlüsselvektor (engl. Key) und den Wertvektor (engl. Value). Anschließend wird die Ähnlichkeit zwischen dem Abfragevektor und dem Schlüsselvektor berechnet, um die Aufmerksamkeitsgewichte zu erhalten. Die Aufmerksamkeitsgewichte geben an, wie wichtig jeder Wertvektor für die Berechnung eines gewichteten Durchschnitts ist. Die gewichteten Wertvektoren werden schließlich summiert, um den Ausgabevektor zu erhalten. Die Self-Attention Mechanismen ermöglichen es den Modellen, komplexe Abhängigkeiten zwischen den Elementen einer Eingabesequenz zu erfassen.

Definition 9 ***Multi-Head Attention*** *beschreibt den Ersatz eines einzelnen Self-Attention-Mechanismus in der Transformer-Architektur durch mehrere parallel arbeitende Self-Attention-Mechanismen.*

Beim Multi-Head Attention-Verfahren werden die Anfrage, der Schlüssel und der Wert h-mal linear auf mehrere Versionen projiziert und parallel berechnet. Die Ausgabe wird dann verkettet und erneut linear projiziert, um den Ausgabevektor zu erhalten. Multi-Head Attention ermöglicht es dem Modell, verschiedene Repräsentationsteilräume der Eingabe an verschiedenen Positionen zu erfassen.

Definition 10 *Ein* ***Feed-Forward-Netz*** *ist eine Art künstliches neuronales Netz, das aus mehreren Schichten besteht und Informationen nur vorwärts von der Eingabe zur Ausgabe fließen lässt.*

Feed-Forward-Netze haben keine zyklischen Verbindungen oder Rückkopplungsschleifen. Sie dienen in Transformern neben den Aufmerksamkeitsnetzen als einzige zweite Komponente der Architektur. Sie enthalten das erlernte Faktenwissen des Modells durch sogenannte Wissensneuronen (Dai u. a., 2022).

2.1.3 *Eigenheiten von Transformer-Modellen*

Definition 11 *Die Bezeichnungen* ***ZeroShot, OneShot*** *und* ***MultiShot*** *beziehen sich auf die Inferenz von Modellen. Die Eingabe enthält hier Beispielaufgaben, bevor die eigentliche Aufgabe gestellt wird.*

Insbesondere bei Question Answering (QA)-Aufgaben und Textgenerierungen werden diese Methoden häufig verwendet. Bei der Beantwortung von Fragen können die Eingaben mehrere Beispielfragen und deren Antworten enthalten, bevor die eigentliche Frage gestellt wird. Dadurch kann das Modell die Formatierung und Art der Aufgabe aus dem Kontext erkennen und bessere Antworten liefern. ZeroShot bedeutet, dass das Modell keine Trainingsbeispiele erhält, um eine bestimmte Aufgabe zu lösen. OneShot bedeutet, dass das Modell nur ein Trainingsbeispiel erhält, um eine bestimmte Aufgabe zu lösen. MultiShot bedeutet, dass das Modell mehrere Trainingsbeispiele erhält.

Definition 12 ***Parameterformate*** *sind Variableneinheiten von lernbaren Parametern. Häufig verwendet werden „Float16“ und „Float32“.*

Sprachmodelle enthalten eine große Anzahl lernbarer Parameter (wie z.B. Gewichte und Bias), sowie Zwischenwerte, die während der Inferenz berechnet werden. Diese Werte können in unterschiedlichen Formaten gespeichert werden. Gängige Formate sind hier „Float16“ und „Float32“. Float steht hier für Floating Value und stellt eine Fließkommazahl dar. Die Zahl hinter Float gibt die Anzahl der Bits an, die

zur Darstellung der Zahl verwendet werden. Eine höhere Anzahl von Bits bedeutet eine höhere Genauigkeit und Wertebereich, aber auch einen höheren Speicherbedarf.

Definition 13 ***Continual Pretraining*** *beschreibt den Prozess des kontinuierlichen, selbstüberwachten Trainings eines vortrainierten Modells.*

Beim Continual Pretraining werden nur die Hyperparameter des Modells angepasst und neue Datensätze verwendet, während auf die bereits erlernten Parameter des Modells aufbauend weitertrainiert wird.

2.1.4 *Eingaben von Transformer-Modellen*

Definition 14 *Ein* ***Token*** *repräsentiert eine Teilmenge eines Wortes und wird in Transformer-Modellen verwendet, um die Eingabe in logische Einheiten zu unterteilen.*

Transformer-Modelle können Eingaben nicht ohne zusätzliche Umwandlung verarbeiten. Neben der Erzeugung von Eingabevektoren muss die Eingabe zunächst in kleinere Einheiten, so genannte Tokens, zerlegt werden. Verschiedene ältere Modelle verwenden dazu Wörter oder Symbolunterteilungen. Dies ist jedoch problematisch.

Durch die Zerlegung der Eingaben in Symbole ist zwar das Vokabular kleiner, welches zu schnelleren Trainingsdurchläufen führt, jedoch muss das Modell vor dem Erlernen von Wortzusammenhängen, Satzstrukturen und Sachverhalten zunächst die Bedeutung der Wörter und deren Zusammensetzung aus Symbolen erlernen. Dadurch geht ein großer Teil der Trainingszeit für das Erlernen der Sprache verloren, was die endgültige Leistungsfähigkeit der Modelle stark einschränkt (Sennrich, Haddow und Birch, 2016).

Definition 15 *Das* ***Vokabular*** *eines Modells ist die Menge aller Token, die das Modell als individuelle Einheiten erkennt. Die Größe des Vokabulars bestimmt die Größe der Input-Embeddings und begrenzt somit die Größe der Eingaben.*

Eine logische Schlussfolgerung wäre hier die Verwendung von Wörtern oder sogar Phrasen als Tokens. Mit zunehmender Größe der Datensätze, die zum Training der Modelle verwendet werden, wächst das Vokabular enorm an. Dies führt zu einer starken Verlangsamung der Trainingsläufe und zu sehr großen Modellen ohne Leistungsvorteil. Wörter mit gleichem Wortstamm oder ähnlicher Bedeutung aufgrund grammatikalischer Regeln (Plural, Genus, Tempus) müssen vom Modell zunächst als „gleiches Wort“ gelernt werden. Daher hat sich die Unterteilung von Wörtern in Teilwörter als Standard durchgesetzt.

2.1.4.1 *Byte Pair Encoding*

Definition 16 ***Byte Pair Encoding*** *ist ein Algorithmus, der iterativ ein Vokabular basierend auf der Häufigkeit von Symbolkombinationen in einem Datensatz aufbaut. Die Vokabulargröße ist der einzige Parameter.*

Sennrich, Haddow und Birch (2016) schlugen zu diesem Zweck die Verwendung von Byte Pair Encoding (BPE) vor. Die Unterteilung von Wörtern in Untergruppen von Wörtern hat bereits zu erheblichen Verbesserungen bei der Übersetzung von Sätzen geführt. Sie hat sich aber auch in anderen Bereichen und Aufgaben wie der Textgenerierung, der Textklassifikation und der Analyse von Emotionen durchgesetzt. Die Unterteilung von Wörtern ist hier eher als das Zusammensetzen von kleinen Teilwörtern zu verstehen. Ausgehend von einem Vokabular, das aus allen Symbolen eines Alphabets besteht, wird dieses durch Zusammenfügen (engl. „Merge“) der Symbole erweitert, deren Kombination im Datensatz am häufigsten vorkommt. Dieser Vorgang wird so lange wiederholt, bis die gewünschte Anzahl von Teilwörtern erreicht ist. Die Anzahl der Teilwörter ist ein Hyperparameter, der je nach Modell und Datensatz variiert.

Die Unterteilung von Wörtern in Teilwörter hat den Vorteil, dass die Größe des Vokabulars nicht mit der Größe des Datensatzes wächst. Dies führt zu einer schnelleren Eingabeverarbeitung und einer besseren Generalisierung der Modelle. Die Unterteilung von Wörtern in Teilwörter hat jedoch auch Nachteile. Sie ist nicht eindeutig, d.h. ein Wort kann in verschiedene Mengen von Teilwörtern zerlegt werden. Dies führt zu einer größeren Anzahl möglicher Eingaben, die das Modell lernen muss.

Ein weiterer Nachteil ist, dass die Zerlegung von Wörtern in Teilwörter nicht immer sinnvoll ist. So kann es vorkommen, dass ein Wort in Teilwörter zerlegt wird, die in der Sprache nicht existieren. Dies wiederum minimiert die Verallgemeinerbarkeit der Modelle. Ein Beispiel hierfür ist das Wort „Datensatz“. Eine sinnvolle Unterteilung wäre hier „Daten“ und „satz“, aber durch den Aufbau des Vokabulars aus den Symbolen des Datensatzes kann es vorkommen, dass das Teilwort „Daten“ nicht die notwendige Häufigkeit besitzt und somit nicht im Vokabular vorhanden ist. Daher muss auch dieses Wort zerlegt werden, z.B. in „Dat“ und „en“. Beide Teilwörter haben in der deutschen Sprache keine Bedeutung, werden aber durch das Modell mit Bedeutung versehen und in Beziehung zu anderen Wörtern gesetzt. Dies führt zu einer unverständlichen Bedeutungsannotation von Teilwörtern und verschlechtert sowohl die Leistung als auch die Nachvollziehbarkeit des Modells und erschwert die Forschung an den Modellen.

2.1.4.2 *Eingabevektoren*

Definition 17 ***Input Embeddings*** *sind eine Darstellung der Eingabedaten in Form von Token in einem dichten Vektorraum.*

Die Repräsentation von Token in einem Vektorraum muss erlernt werden und ermöglicht es, Token, die semantische Ähnlichkeiten oder Beziehungen zueinander aufweisen, geometrisch nahe beieinander zu platzieren. Dadurch ist nicht nur das Token selbst, sondern auch die Beziehung zu anderen Tokens bekannt. Mit zunehmender Größe des Vokabulars müssen auch die Dimensionen der Input Embeddings vergrößert werden, da es sonst zu ungewollten Kollisionen zwischen Tokens kommen kann.

Definition 18 ***Positional Encodings*** *stellen sogenannte relative und absolute Positionen der Token in der Eingabe dar.*

Relative und absolute Positionen von Token innerhalb einer Eingabe werden in einem Vektorraum dargestellt und ermöglichen es dem Modell, die Positionen der Token zu berücksichtigen. Positional Encodings können gelernt oder berechnet werden und werden nur in Kombination mit Input Embeddings verwendet.

2.2 NEURONALE NETZE

Um die Architektur des Transformers zu verstehen, ist es notwendig, die Grundlagen neuronaler Netze zu kennen. Neuronale Netze wurden erstmals 1943 von McCulloch und Pitts (1943) beschrieben und haben seitdem viele Veränderungen und Verbesserungen erfahren. Eine Beschreibung der grundlegenden Techniken findet sich in Kinnebrock (1994). Aufbauend auf diesem Buch wird im Folgenden ein Überblick über die Funktionsweise von Neuronalen Netzen gegeben.

2.2.1 *Architektur*

Neuronale Netze ahmen die Funktionsweise des menschlichen Gehirns mit Hilfe mathematischer Gleichungen nach. Ein Netz besteht aus einer Anzahl von Neuronen, die sowohl Eingangs- als auch Ausgangsverbindungen zu anderen Neuronen haben. Neuronen haben die Eigenschaft, von bestimmten Eingängen einen Impuls an die Ausgänge zu senden, wenn ein Schwellenwert überschritten wird. Im Gegensatz zum Gehirn sind normale neuronale Netze jedoch logischer aufgebaut. Die Neuronen sind in Schichten angeordnet, wobei die erste Schicht die Eingangsschicht mit den Eingangsneuronen E_N und die letzte Schicht die Ausgangsschicht mit den Ausgangsneuronen A_N ist. Die Schichten zwischen diesen beiden Grenzen werden

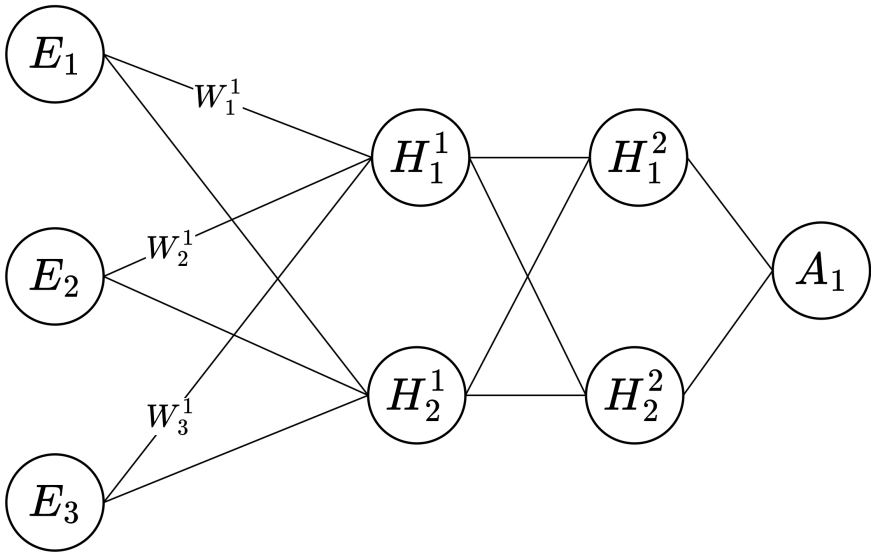

Abbildung 2.2: Beispiel-Aufbau eines Neuronalen Netzes mit 3 Eingangsneuronen, 4 versteckten Neuronen und einem Ausgangsneuronen.

als versteckte Schichten (engl. „Hidden Layers“) mit Neuronen H_N^L bezeichnet.

Abb. 2.2 zeigt eine Beispielkonfiguration mit 3 E_N, hier bezeichnet als E_1, E_2, E_3, zwei versteckten Schichten mit je zwei H_N^L, hier bezeichnet als H_1^1, H_2^1 und H_1^2, H_2^2 und einem Ausgangsneuron A_1. Die Neuronen sind mit sogenannten Gewichten W_N verbunden. Man spricht hier von einer Vorwärtsverbindung. Das bedeutet, dass die Eingaben durch die Schichten in Richtung Ausgabe weitergereicht werden, aber nicht zurück zur Eingabe fließen können. Im Beispiel in Abb. 2.2 sind diese Verbindungen vollständig. Jedes Neuron einer Schicht hat eine Verbindung zu jedem Neuron der nächsten Schicht. Hier nicht gezeigt, hat jedes Neuron auch eine Eingangsverbindung von einem Bias B_N.

2.2.2 *Funktionsweise*

Die Anwendung des Neuronalen Netzes erfolgt in zwei Schritten. Zuerst wird eine Eingabe in die Eingangsneuronen E_N eingegeben. Diese Eingabe wird durch die Schichten weitergegeben, bis sie in der Ausgabeschicht A_N ankommt. Jetzt muss das Ergebnis interpretiert werden. Dieser Vorgang wird auch als Inferenz bezeichnet.

Definition 19 ***Inferenz*** *beschreibt den Prozess, bei dem ein Modell aufgrund der gelernten Parameter eine Vorhersage trifft.*

Ein Anwendungsbeispiel wäre die Erkennung von handgeschriebenen Zahlen. Die Eingabe wäre hier ein Bild der Ziffer, das in ein

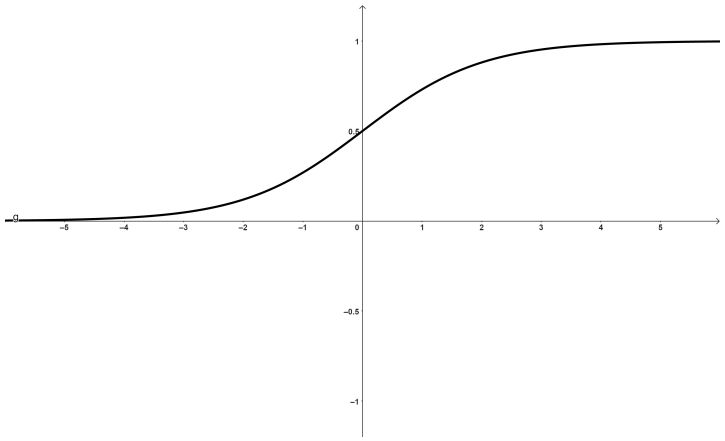

Abbildung 2.3: Die Sigmoid-Funktion $f(x) = \frac{1}{1+e^{-x}}$

neuronales Netz eingespeist wird. Jedes Neuron repräsentiert den Grauwert eines Pixels im Bereich $[0;1]$. Die Ausgabe stellt die erkannte Ziffer dar. Für jede mögliche Ziffer (0-9) gibt es ein Ausgangsneuron, das vom neuronalen Netz einen Wert zwischen 0 und 1 erhält. Das Neuron mit dem höchsten Ausgang repräsentiert die erkannte Ziffer.

Ein Berechnungsbeispiel für das erste Neuron der ersten versteckten Schicht H_1^1 sieht wie folgt aus:

$$H_1^1 = ReLU(W_1^1 \cdot E_1 + W_2^1 \cdot E_2 + W_3^1 \cdot E_3 + B_1^1) \tag{2.1}$$

W_N^1 stellen hier die Gewichte der ersten Schicht dar. Gewichte sind Parameter, die während des Trainings eines Netzes angepasst werden, um eine optimale Ausgabe von A_N zu erhalten. Diese Parameter werden auch als lernbare Parameter bezeichnet.

Definition 20 *Ein **Lernbarer Parameter** ist ein Parameter, der zufällig initialisiert wird und während des Trainings angepasst wird.*

B_N^1 ist der Bias und ebenfalls ein lernbarer Parameter. Alle Eingabewerte werden gemäß der Formel aufsummiert und mit Hilfe einer Aktivierungsfunktion transformiert. Diese Aktivierungsfunktion war in früheren Netzen die Sigmoidfunktion (Abb. 2.3). In modernen Netzen und auch in Transformer-Modellen wird hier die ReLU-Funktion (Abb. 2.4) verwendet.

Definition 21 *Die **Aktivierungsfunktion** weißt jedem potentiellen Wert eines Neurons einen Wert in einem vorgegebenem Intervall zu. Sie ist nichtlinear und verhindert somit die Reduzierung eines Neuronalen Netzes auf eine lineare Funktion.*

Üblicherweise stellt eine Aktivierungsfunktion eine Transformation der Eingabewerte in einen nichtlinearen Ausgabewert zwischen 0 und

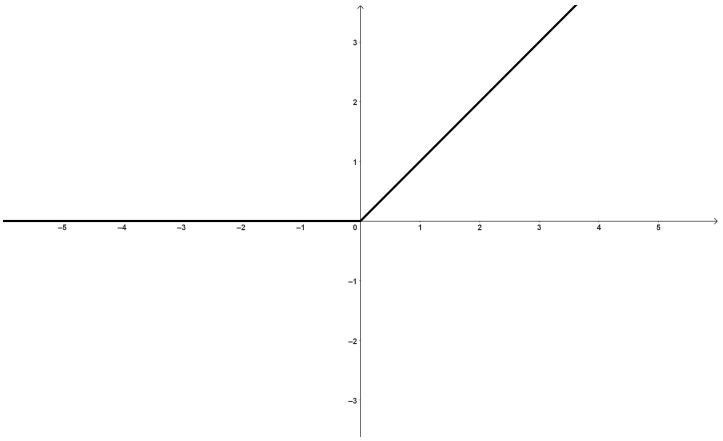

Abbildung 2.4: Die ReLU-Funktion $f(x) = \max(0, x)$

1 dar. Dies ist jedoch nicht zwingend, da z.B. die ReLU-Funktion Werte zwischen 0 und ∞ ausgibt, aber schneller berechnet werden kann als die Sigmoidfunktion. Aktivierungsfunktionen sind notwendig, um die Komplexität eines neuronalen Netzes zu erhöhen und damit seine Fähigkeit, komplexere Probleme zu lösen, zu ermöglichen. Ohne Aktivierungsfunktion würden A_N Neuronen eine Linearkombination der Eingabewerte erzeugen, wodurch nur lineare Probleme gelöst werden könnten. Ein einfaches Beispiel ist die Berechnung der XOR-Funktion. Die XOR-Funktion verknüpft zwei Eingaben zu einer Ausgabe. Wenn eine der beiden Eingaben 1 ist, soll auch die Ausgabe 1 sein. Andernfalls soll die Ausgabe 0 sein. Diese einfache Funktion kann ohne eine Aktivierungsfunktion nicht gelöst werden.

Die Ausgabewerte A_N sind ebenfalls Werte im Bereich der Aktivierungsfunktion und nicht immer repräsentativ für das Problem, das das neuronale Netz lösen soll. Daher werden diese Werte interpretiert und stellen häufig einen Konfidenzwert für eine bestimmte Klasse dar. Wenn das Neuron A_1 den Maximalwert 1 hat, dann ist sich das Netz zu 100 % sicher, dass die Eingabe der Klasse 1 entspricht. Bei Transformern ist die Ausgabe A_N ein Vektor über das gesamte Vokabular des Transformers und gibt an, inwieweit das Modell davon ausgeht, dass das jeweilige Token dem Eingabetext folgt.

2.2.3 *Training*

Bevor ein neuronales Netz korrekte Ergebnisse liefern kann, muss es trainiert werden. Training bedeutet die iterative Anpassung der lernbaren Parameter, um eine möglichst gute Ausgabe zu erhalten. Eine gute Ausgabe wird durch eine Fehlerfunktion beschrieben. Diese Fehlerfunktion gibt einen einzelnen Wert zurück, der den Eingaben in das neuronale Netz entspricht und umgekehrt proportional zur

guten Ausgabe ist. Geometrisch stellt eine Fehlerfunktion eine mehrdimensionale Fläche dar, wobei jeder lernbare Parameter auf einer Achse liegt und das Minimum dieser Fläche ein Optimum darstellt. Dieses Optimum ist das Trainingsziel. Kleine Änderungen einzelner Parameter verändern die Ausgabe der Fehlerfunktion und können somit zeigen, ob diese Änderung zu einer Verbesserung oder Verschlechterung des neuronalen Netzes geführt hat. Hier besteht bereits die Unterscheidung zwischen einem überwachten, selbstüberwachten und unüberwachten Lernverfahren.

Definition 22 *Bei dem **Überwachten Lernen** sind sowohl die Eingabe als auch die erwartete Ausgabe während des Trainings bekannt. Die verwendete Fehlerfunktion ergibt sich aus der Differenz zwischen erwarteter und tatsächlicher Ausgabe.*

Definition 23 *Bei dem **Unüberwachten Lernen** ist während des Trainings nur die Eingabe bekannt. Die Fehlerfunktion muss aus dem Kontext berechnet oder abgeleitet werden.*

Definition 24 *Bei dem **selbstüberwachten Lernen** ist während des Trainings nur die Eingabe bekannt. Die Ausgabe entspricht jedoch einem Teil der Eingabe, so dass die Fehlerfunktion berechnet werden kann.*

Ein Beispiel für überwachtes Lernen ist die Klassifizierung von Bildern. Der Datensatz enthält sowohl die Bilder als auch die zugehörigen Klassen. Daher ist die erwartete Ausgabe des Modells für ein Bild genau die Klasse. Transformer-Modelle sind selbstüberwachende Systeme. Hier ergibt die Eingabe die erwartete Ausgabe, da das folgende Token der Eingabe als korrekt angesehen wird. Unüberwachte Systeme sind z.B. Agenten, die Spiele lernen. Hier gibt es in den meisten Fällen eine Belohnungsfunktion, die angibt, wie gut das Modell gespielt hat. Belohnungsfunktionen können beschreiben, wie weit das Auto im Spiel gefahren ist oder wie viele Punkte gesammelt wurden.

2.2.3.1 *Backpropagation*

Definition 25 ***Backpropagation** beschreibt den Prozess der iterativen Anpassung von Gewichten und Bias auf Basis des Gradienten einer Fehlerfunktion.*

Eine der wichtigsten Methoden zur iterativen Anpassung von Gewichten und Bias und damit zum Training neuronaler Netze ist die Methode der Fehlerrückführung (engl. Backpropagation). Diese Gradientenabstiegsmethode wurde erstmals 1986 von Rumelhart, Hinton und Williams (1986) beschrieben und ist ausführlich in Kinnebrock (1994) erklärt. Um den Rahmen dieser Arbeit nicht zu sprengen,

wird hier auf eine mathematische Beschreibung verzichtet und nur die Grundidee beschrieben.

Die Fehlerfunktion C ist eine Funktion aller Gewichte und Bias des neuronalen Netzes und seiner Ausgabe. Ihr Minimum zu finden bedeutet, ein Optimum für das Modell zu finden. Zu Beginn des Trainings werden alle Gewichte mit Zufallszahlen initialisiert. Nun wird die Ausgabe des Modells berechnet und mit der erwarteten Ausgabe verglichen. Die Funktion C hat einen Gradienten ΔC, der die Richtung des steilsten Anstiegs der Funktion beschreibt. Soll die Funktion minimiert werden, so wird der Gradient in die entgegengesetzte Richtung angewendet. Dieser Gradient setzt sich aus partiellen Ableitungen der Fehlerfunktion über alle lernbaren Parameter zusammen.

LERNRATE

Definition 26 *Die **Lernrate** entspricht der Schrittweite des Gradientenabstiegs. Höhere Lernraten führen zu schnelleren Lernprozessen, können aber auch zu Oszillationen führen. Kleinere Lernraten führen zu langsameren Lernvorgängen und konvergieren möglicherweise nicht zu einem Minimum.*

Der Prozess des „Voranschreiten zum Minimum“ bedeutet die Anpassung der lernbaren Parameter gemäß den partiellen Ableitungen. Die Anpassung der Parameter ist ein Hyperparameter des Trainings und wird als Lernrate (engl. „Learning Rate“) bezeichnet.

Definition 27 ***Hyperparameter** sind Parameter, die Eigenschaften oder Techniken des Modells repräsentieren und vor dem Training festgelegt werden. Sie können vom neuronalen Netz nicht gelernt werden.*

Wird die Lernrate zu hoch gewählt, besteht die Gefahr, dass ein Minimum der Fehlerfunktion übersprungen wird und um diesen Punkt oszilliert. Wählt man die Lernrate zu niedrig, so dauert das Training des neuronalen Netzes zu lange, um das Minimum zu erreichen. Bei Transformern und tiefen neuronalen Netzen ist die Lernrate nicht konstant, sondern wird während des Trainingsprozesses regelmäßig angepasst. Sehr große Modelle haben oft mehrere Milliarden Parameter und benötigen daher eine sehr kleine Lernrate, um nicht zu oszillieren. Zu Beginn eines Trainings passt sich das Modell mit einer höheren Lernrate zu stark an kleine Änderungen an und findet ein lokales Minimum, das weit über dem globalen Minimum liegt. Daher wird eine Aufwärmphase verwendet, um die Lernrate langsam zu erhöhen und das Modell an die Daten anzupassen.

Nach der Aufwärmphase wird die Lernrate langsam reduziert, so dass zu Beginn große Anpassungen der Parameter möglich sind und

gegen Ende des Trainings nur noch eine Feinanpassung des gefundenen Minimums möglich ist.

STOCHASTISCHER GRADIENTENABSTIEG

Die Berechnung des Gradienten der Fehlerfunktion über alle lernbaren Parameter ist nur sinnvoll, wenn sie über den gesamten Datensatz berechnet wird. Jede Eingabe des Datensatzes verändert die Ausgabe des Modells und somit auch die Fehlerfunktion. Das arithmetische Mittel der Fehlerfunktionen über alle Eingaben des Datensatzes ist die Fehlerfunktion des gesamten Datensatzes.

$$C = \frac{1}{n} \sum_{k=0}^{n-1} C_k \tag{2.2}$$

Hierbei ist n die Anzahl der Eingaben des Datensatzes und C_k die Fehlerfunktion der k-ten Eingabe. Die Berechnung des Gradienten über alle Eingaben des Datensatzes ist jedoch sehr rechenintensiv und wird daher durch eine Stichprobe des Datensatzes ersetzt. Diese Stichprobe wird als Batch bezeichnet.

Definition 28 *Ein **Batch** repräsentiert eine Teilmenge von Eingaben, welche geschlossen verarbeitet werden, bevor die Gewichte und Bias angepasst werden.*

Die Größe des Batches ist ein weiterer Hyperparameter des Trainings. Die Berechnung des Gradienten über einen Batch wird als Stochastic Gradient Descent bezeichnet, da die Stichprobe des Datensatzes nur eine Approximation des Gradienten ist. Dies bedeutet, dass die Richtung des Gradienten nur näherungsweise dem tatsächlichen Gradienten entspricht, die Berechnung aber wesentlich schneller ist. Die Größe des Batches ist ein Kompromiss zwischen der Genauigkeit des Gradienten und der Geschwindigkeit der Berechnung.

OVERFITTING

Definition 29 ***Overfitting** beschreibt den Prozess, bei dem ein Modell die Trainingsdaten auswendig lernt und nicht mehr in der Lage ist, neue Daten korrekt zu klassifizieren. Overfitting kann durch einen Leistungsabfall zwischen Trainings- und Testdaten erkannt werden.*

Ein logisches Ende des Trainings ist erreicht, wenn sich die Fehlerfunktion über einen bestimmten Zeitraum nicht mehr ändert und somit ein lokales Minimum gefunden wurde. Dieses lokale Minimum ist jedoch nicht immer erwünscht. Modelle, die den Datensatz auswendig lernen, erreichen einen sehr guten Wert für die Fehlerfunktion,

der Prozess der Backpropagation wirkt unterstützend. Diese Modelle sind jedoch nicht in der Lage, neue Daten korrekt zu klassifizieren und sind daher für die Praxis ungeeignet. Dieser Prozess des Auswendiglernens wird als Overfitting bezeichnet.

Overfitting kann durch verschiedene Methoden verhindert werden. Um Overfitting zu erkennen, wird der Datensatz in Trainings- und Testdaten aufgeteilt. Es wird nur auf den Trainingsdaten trainiert und mit den Testdaten verglichen. So kann festgestellt werden, wie gut das Modell auf Daten generalisiert, die es noch nicht gesehen hat. Nähert sich die Fehlerfunktion bei den Testdaten einem Minimum, kann das Training beendet werden, da weitere Parameteranpassungen nur zu einem Auswendiglernen des Trainingsdatensatzes führen.

2.3 DATENVERARBEITUNG

Um ein Modell trainieren zu können, müssen die verwendeten Datensätze in eine für das Modell verständliche Form gebracht werden.

Definition 30 ***Datenkuration*** *beschreibt den Prozess der Datenaufbereitung, um diese für die Verarbeitung durch ein Modell vorzubereiten.*

Die Datenkuration ist ein wichtiger Schritt dieser Arbeit und wird weiter in Abschnitt 4.2 genauer beschrieben. Da Modelle nicht mit normalen Textdateien wie Word oder PDF umgehen können, müssen diese in eine für das Modell verständliche Form umgewandelt werden.

2.3.1 *Glossar der Daten*

In dieser Arbeit geht es hauptsächlich um Daten und die darin enthaltenen Informationen und das Wissen. Zum besseren Verständnis dieser Unterscheidung folgt hier die Definition nach Winter u. a. (2023).

Definition 31 ***Wissen*** *sind generelle Informationen über Konzepte in einer bestimmten Domäne.*

Im Kontext dieser Arbeit stellt Wissen Informationen aus Winter u. a. (2023) dar. Diese Informationen müssen von einem Modell gelernt werden, um das darin enthaltene Wissen zu reproduzieren.

Definition 32 ***Informationen*** *sind spezifische Festlegungen über Entitäten wie zum Beispiel Fakten, Dinge, Personen, Prozesse, Ideen, Konzepte oder Ereignisse.*

Eine Information ist unabhängig von ihrer Darstellung und ist das grundlegende Ziel dieser Arbeit. Informationen können unterschiedlich formuliert sein, aber den gleichen Informationsgehalt enthalten.

Daher wird, wie in Abschnitt 4.4 beschrieben, eine Antwort auf eine Frage, die die richtige Information enthält, unabhängig von ihrer Darstellung als richtig angesehen.

Definition 33 ***Daten*** *sind Informationen oder Wissen in einer strukturierten Form, die für die Verarbeitung, Kommunikation oder Interpretation von Menschen oder Maschinen geeignet sind.*

Die in dieser Arbeit verwendeten Daten sind Eingabedaten wie das Buch von Winter u. a. (2023) und die Fragen zu diesem Buch, Ausgabedaten aus dem trainierten Modell als Antworten auf die Fragen und Daten, die im Modell enthalten sind, aber nicht in einer für Menschen verständlichen Form vorliegen.

3 STAND DER FORSCHUNG

In diesem Kapitel werden aktuelle Artikel und Forschungsergebnisse zu den Themen Sprachmodelle, Continual Pretraining und Problemen bei der Nutzung von Sprachmodellen vorgestellt.

3.1 CONTINUAL PRETRAINING UND DIE NUTZUNG VON SPRACHMODELLEN

Ein Transformer-Modell als Wissensbasis wird in Omar u. a. (2023) mit verschiedenen State of the Art (Stand der Technik) (SOTA)-Modellen verglichen. Sie zeigen eine deutliche Verbesserung der Robustheit gegenüber Eingabefehlern, der Erklärbarkeit von Antworten und des Fragenverständnisses bei komplexeren Fragen mit mehreren Fakten durch ChatGPT, zeigen aber Probleme bei der Aktualität von Wissen, dem Wissen über spezifische Domänen und vor allem bei der korrekten Beantwortung von Fragen. Der Grund dafür ist eine grundlegende Eigenschaft von GPT-Modellen, nämlich die fehlende Inkorporation aktuellen Wissens. Das Training von GPT-Modellen ist ein aufwendiger Prozess und kann nicht bei jeder Inferenz (Nutzung des Modells durch Generierung von Text) durchgeführt werden. Darüber hinaus wurde ChatGPT ohne zusätzliches Continual Pretraining eingesetzt. Eine Anpassung an die Domänen und eine Verbesserung der Korrektheit der Antworten stehen daher noch aus.

Radford und Narasimhan (2018) zeigen die Verbesserung der Leistung von Transformer-Modellen durch generatives Pretraining und eine weitere Verbesserung durch überwachtes Finetuning. Auch hier ist ein klarer Trend erkennbar. Die Ergebnisse der Modelle verbessern sich mit zunehmender Größe des Datensatzes, mit zunehmender Länge des Trainingsprozesses und mit zunehmender Größe der Modelle.

Diese Tendenz wird durch Kaplan u. a. (2020) bestätigt, die hier den Einfluss verschiedener Einflussgrößen auf die Gesamtleistung eines Modells berechnen. Die hier verwendeten Einflussgrößen erlauben eine Vorhersage der Leistung eines Modells. Der Artikel schließt mit einer Abschätzung der theoretischen Maximalleistung und damit der Maximalgröße von Transformer-Modellen.

Um den beschriebenen Skalierungsregeln zu folgen, aber die Trainingszeit und die benötigte Datenmenge zu reduzieren, gibt es die Möglichkeit des Continual Pretraining. Gururangan u. a. (2020) hat

diese Methodik angewandt und gezeigt, dass Modelle immens davon profitieren, domänenspezifisches Wissen zu adaptieren und aus der großen Menge an Basisdaten bessere korrekte Antworten in einer spezifischen Domäne zu generieren. Erstmals in Lee u. a. (2019) eingesetzt, um das Basismodell Bidirectional Encoder Representations from Transformers (BERT) an die biomedizinische Domäne anzupassen, erweiterten Gururangan u. a. (2020) diese Methode und zeigten ihre Anwendbarkeit auf verschiedene Domänen und Aufgaben. Continual Pretraining auf aufgabenspezifischen Daten verbessert wiederum die Leistung für spezifische Aufgaben, während die Trainingszeit um den Faktor 60 kürzer ist im Vergleich zu Continual Pretraining auf Domänen. Eine Kombination beider Arten liefert die besten Ergebnisse.

Aber nicht nur das Continual Pretraining verbessert die Ausgabe der Modelle, sondern auch das überwachte Finetuning. Ziegler u. a. (2019) beschreiben in ihrem Artikel die Effektivität von Reinforcement Learning als Finetuning-Methode zur Lösung von Aufgaben der Weiterführung und Zusammenfassung von Texten. Finetuning benötigt jedoch gekennzeichnete Daten (engl. „labeled data“, Daten mit bekannten korrekten Ausgaben), die in der Regel aufwendig zu erstellen sind und nicht immer in der benötigten Menge zur Verfügung stehen. In dieser Arbeit wird aufgrund der mangelnden Verfügbarkeit von gekennzeichneten Daten auf ein Finetuning verzichtet.

3.2 AKTUELLE MODELLE UND DEREN NUTZBARKEIT

Die Erkenntnis, dass die Leistung von Modellen mit zunehmender Größe, Trainingszeit und Datenmenge steigt, hat zur Entwicklung einer Reihe von Modellen mit unterschiedlichen Architekturen, Anwendungsfällen und Leistungen geführt. Ein erster Durchbruch in der Leistungsfähigkeit von Transformer-Modellen wurde von Radford u. a. (2019) mit dem Modell GPT-2 erzielt. Im Vergleich zum ersten veröffentlichten GPT-1 Modell (Radford und Narasimhan, 2018) zeigten sie, dass Sprachmodelle Aufgaben lösen können, ohne explizit überwacht zu werden. Ebenso stellten sie fest, dass die Größe eines Modells, sei es hier die Anzahl der Parameter, die Größe des Datensatzes oder die Länge des Trainings, eine grundlegende Notwendigkeit für den erfolgreichen Einsatz der ZeroShot-Methode ist. Bereits hier konnten sie ohne jegliche Änderung der Architektur im ZeroShot Rahmen je nach Aufgabenstellung erfolgreiche, SOTA-kompetitive Ergebnisse erzielen.

OpenAI gelang mit GPT-3 ein Durchbruch in der Popularität und beschrieben ihren Ansatz in Brown u. a. (2020). In ihrem Artikel demonstrierten sie die Leistungssteigerung durch größere Modelle und zeigten, dass diese Leistung auch ohne Finetuning erreicht werden

kann. Sie verglichen auch das Antwortverhalten in Abhängigkeit von FewShot- und ZeroShot-Eingaben, wobei erstere bessere Ergebnisse lieferten. Diese Ergebnisse unterstützen die Hypothese, dass das in dieser Arbeit verwendete Modell auch ohne Finetuning eine gute Leistung erzielen kann.

Weiter im Jahr 2023 veröffentlichte OpenAI GPT-4 und stellten es in OpenAI (2023) vor. Neben deutlich besseren Ergebnissen durch ein noch größeres Modell mit mehr Parametern gelang es nun auch, Bilddaten als Eingabe zu verarbeiten. Dieser Artikel unterstreicht erneut die Annahme, dass größere Modelle eine bessere Leistung und ein besseres Verständnis der natürlichen Sprache haben. Eine Verwendung dieses Modells sowie des Modells GPT-3 ist nicht möglich, da diese Modelle derzeit nicht veröffentlicht sind.

Im Gegensatz zu den Modellen, die von OpenAI publiziert wurden, veröffentlichten Black u. a. (2022) GPT-NeoX. Ein Modell, das in Größe und Leistungsfähigkeit GPT-3 ähnelt, jedoch auf der Architektur von GPT-J[1] basiert und im Open Source Rahmen veröffentlicht wurde. Sie zeigten, dass die meisten interessanten Fähigkeiten eines Modells erst ab einer bestimmten Anzahl von Parametern sichtbar werden.

Zuletzt wurden von Touvron u. a. (2023a) die Large Language Model Meta AI (LLaMA)-Modelle in verschiedenen Größen veröffentlicht. Ein klarer Vorteil gegenüber anderen Modellen in ihrer Anwendbarkeit ist hier der Fokus auf eine längere Trainingszeit und einen größeren Datensatz gegenüber der Modellgröße. Sie zeigten in fast allen Aufgabenbereichen bessere Ergebnisse als andere Modelle wie GPT-3 und Pathways Language Model (PaLM) mit deutlich weniger Parametern. Damit sind diese Modelle billiger, einfacher und schneller im Training und in der Anwendung mit gleichen oder besseren Ergebnissen. Diese Modelle wurden ebenfalls veröffentlicht und sind daher für diese Arbeit verfügbar.

3.3 FORSCHUNG UND PROBLEME VON MODELLEN

Neben der Entwicklung neuer Modelle wurden auch neue Ansätze zur Verbesserung des Continual Pretraining und der Adaption von Modellen entwickelt. Pfeiffer u. a. (2020) stellten in ihrem Artikel Adapter vor, die es ermöglichen, zusätzliche NNs auf verschiedenen Ebenen der Transformer-Architektur einzusetzen. Mit ihrer Hilfe kann die Adaption an andere Aufgaben und Domänen erreicht werden, ohne dass das gesamte Modell kontinuierlich vortrainiert werden muss, da während des Trainings alle Parameter des Ausgangsmodells fixiert bleiben,

1 Ben Wang `https://github.com/kingoflolz/mesh-transformer-jax` (abgerufen am 3.6.2023)

während die neu eingefügten Adapter trainiert werden. Darüber hinaus können bereits vortrainierte Adapter zu weiteren Domänen und Aufgaben in bestehende Modelle eingefügt werden, ohne dass ein erneutes Training erforderlich ist. Das veröffentlichte System basiert auf dem Artikel von Houlsby u. a. (2019), in dem die Autoren das BERT-Modell auf 26 verschiedene Natural Language Processing (NLP)-Aufgaben trainierten, mit einer Anpassung von nur 3,6 % der Parameter und einer Leistungsminimierung von 0,4 % (ein ursprüngliches Training der Modelle auf diese Aufgaben hätte 100 % aller Parameter angepasst). Sie bewiesen damit die Effizienz dieses Ansatzes, ohne die Leistung wesentlich zu beeinträchtigen.

Dai u. a. (2022) untersuchten die Fähigkeiten von Large Language Model (LLM)s im Hinblick auf ihre Fähigkeit, faktisches Wissen wiedergeben zu können, ohne eine Datenbank mit Fakten als Grundlage während des Betriebs zur Verfügung zu haben. Sie stellten fest, dass vor allem in tieferen Ebenen die neuronalen Netze so genannte „Wissensneuronen“ besitzen, die mit bestimmten Fakten korrelieren. Diese Wissensneuronen werden aktiv, wenn ein bestimmter Fakt in der Eingabe angesprochen wird und können durch Verstärkung oder Unterdrückung dazu führen, dass das Modell diesen Fakt besser berücksichtigt oder „vergisst“.

Die Untersuchung von Modellen und ihrer Fähigkeit, Sachverhalte zu erlernen und zu reproduzieren, wurde erstmals von Petroni u. a. (2019) vorgestellt. Die hier verwendeten Modelle BERT und Embeddings from Language Models (ELMo) wurden auf ihr Potential als unüberwachte offene Domäne QAS untersucht und zeigten gute Ergebnisse im Vergleich zu anderen SOTA-Systemen. Mehrsprachige Modelle wurden von Jiang u. a. (2020) auf die gleichen Eigenschaften hin untersucht, schnitten jedoch deutlich schlechter ab. Diese Ergebnisse deuten darauf hin, dass ein mehrsprachiges Modell für den Einsatz als QAS deutlich schlechter geeignet ist, da ein Großteil der Leistung dieser Modelle in das Verstehen von Übersetzungen in andere Sprachen fließt.

Neben den großen Erfolgen der neuen Modelle treten jedoch auch neue Probleme bei der Anwendung dieser Modelle auf. Neben Falschaussagen ergeben sich Probleme durch soziale und andere Biases in den Antworten, Selbstüberschätzung bei Falschaussagen, die wiederum zu schwerwiegenden Problemen in der Anwendung dieser Modelle führen können, Generierung von schädlichen Inhalten, Unterstützung von Kriminalität durch Expertise und andere Probleme. Eine Analyse der Ergebnisse dieser Arbeit in Bezug auf die ethischen Richtlinien der GMDS findet sich in Abschnitt 1.5. OpenAI (2023) widmeten einen eigenen Abschnitt ihres Artikels der Untersuchung dieser Probleme

und ihrer Adressierung. Sie zeigen darin grundsätzliche Probleme bei der Anwendung von Sprachmodellen auf, halten sich aber mit konkreten Lösungsvorschlägen zurück.

In Dehouche (2021) beschreibt der Autor weitere Fragen zum Umgang mit Antworten von Sprachmodellen und deren Konflikt mit dem Urheberrecht. Wem gehört der generierte Text - den Autoren der Datensätze, auf denen das Modell trainiert wurde, der Firma, der das Modell gehört, dem Benutzer, der das Modell anleitet? Auch hier zeigen sich ungelöste Probleme in der Anwendung von Sprachmodellen und bieten Raum für weitere Forschung. Eine abschließende Antwort auf diese Fragen gibt es noch nicht, so dass die Verwendung eines Sprachmodells zum Zeitpunkt dieser Arbeit nur von der Lizenzierung des jeweiligen Modells durch die Autoren des Modells und der Lizenzierung der Datensätze, die für das Continual Pretraining verwendet werden, abhängt. Das hier verwendete Buch Winter u. a. (2023) steht unter einer Open-Access-Lizenz und kann daher uneingeschränkt für ein Continual Pretraining verwendet werden.

4 LÖSUNGSANSATZ

Diese Arbeit beschäftigt sich mit der Beantwortung von Wissensfragen zu Winter u. a. (2023) mit Hilfe eines vortrainierten Sprachmodells, das unter Verwendung des Buches Winter u. a. (2023) auf diese Domäne trainiert wird. Die Aufgabenstellung unterteilt sich in fünf Teile:

1. Die Auswahl eines Sprachmodells, welches die Aufgabe lösen kann.
2. Die Kuration und Umwandlung des Buches von Winter u. a. (2023) in eine von dem Modell verständliche Form.
3. Die Durchführung des Trainings des ausgewählten Modells.
4. Die Erstellung eines Evaluierungsdatensatzes auf Basis von Klausuren, welche inhaltlich auf dem Buch von Winter u. a. (2023) basieren.
5. Die Evaluierung des Modells mit Hilfe des erstellten Evaluierungsdatensatzes.

4.1 AUSWAHL VON SPRACHMODELLEN

Für die Auswahl eines Sprachmodells müssen verschiedene Kriterien erfüllt sein. Ausgehend von der Aufgabenstellung in Abschnitt 1.4 stehen verschiedene Modelle der Transformer-Architektur zur Auswahl. Die Auswahl beschränkt sich auf Decoder-basierte Modelle, da diese die Eigenschaften eines autoregressiven Modells besitzen. Wie bereits in Abschnitt 2.1.1 beschrieben, ist das Ziel die Generierung von Text auf Basis einer Fragestellung. Diese Fragestellung wird zu Beginn als Eingabe zur Verfügung gestellt, woraufhin das Modell kontinuierlich neue Tokens generiert. Neu generierte Tokens werden zusammen mit der ursprünglichen Frage als Eingabe verwendet, um den nächsten Token zu generieren. Dieses Verhalten entspricht autoregressiven Modellen und erfordert Decoder-basierte Modelle. Ein Decoder-basiertes Modell verwendet sowohl Encoder als auch Decoder-Schichten um Eingaben zu verstehen und daraus Text zu generieren. Encoder-basierte Modelle besitzen in der Regel keine Decoder-Schichten und sind nur für die Textvervollständigung oder Textklassifikation gedacht, aber nicht für die Textgenerierung.

Zusätzlich sind folgende Eigenschaften erwünscht. Das Modell soll eine gute ZeroShot-Performance haben und nicht nur eine gute FewShot- oder OneShot-Performance. Diese Arbeit untersucht die

Modell	Verfügbarkeit	ZeroShot	Größe	Training	Inferenz	Inferenz (4Bit)
GPT-2	frei	schlecht	1,5	24 GB	6 GB	750 MB
GPT-3	nur Inferenz	gut	175	1,4 TB	350 GB	43,8 GB
GPT-4	nur Inferenz	sehr gut	?	?	?	?
GPT-J	frei	gut	6,7	104 GB	26 GB	3,4 GB
GPT-NeoX	frei	gut	20	160 GB	40 GB	10 GB
LLaMA	limitiert	gut	7	56 GB	14 GB	3,5 GB
	limitiert	gut	13	104 GB	23 GB	5,8 GB
	limitiert	sehr gut	33	264 GB	66 GB	16,5 GB
	limitiert	sehr gut	65	520 GB	130 GB	32,5 GB
Llama 2	limitiert	gut	7	56 GB	14 GB	3,5 GB
	limitiert	gut	13	104 GB	23 GB	5,8 GB
	limitiert	sehr gut	70	560 GB	140 GB	35 GB

Tabelle 4.1: Übersicht über die zur Auswahl stehenden Modelle. Größe ist in Milliarden Parameter angegeben.

Fähigkeit eines Modells, Wissen aus domänenspezifischer Literatur (hier aus Winter u. a. (2023)) zu reproduzieren. Indem frühere Fragen und Antworten als Kontext verwendet werden, werden die Ergebnisse synthetisch durch zusätzliches Wissen im Kontext verbessert. Einige Fragen könnten nicht durch das Modell mit Hilfe des gelernten Wissens beantwortet werden, sondern durch das Wissen im Kontext. Das Modell soll auch ohne Finetuning eine gute Leistung erbringen. Da ein Finetuning in dieser Arbeit aufgrund der fehlenden Datenmenge nicht möglich ist, soll ein Modell gewählt werden, das auch ohne dieses Finetuning eine gute Performance erreicht.

Hier zur Auswahl stehende Modelle sind GPT-2, GPT-3, GPT-4, GPT-J, GPT-NeoX, LLaMA und Llama 2 gezeigt in Tabelle 4.1. Die angegebenen Speichergrößen beziehen sich hier auf die notwendige Speichermenge zum Training des Modells und der Inferenz. Berechnet wird die Speicheranforderung zum Training mit der Gleichung (4.1).

$$\text{Training Speicherbedarf} = \text{Anzahl Parameter} \cdot \frac{\text{Parameterformat}}{8} \cdot 4 \tag{4.1}$$

Sprachmodelle benutzen die Parameterformate FP16 oder FP32, welche 2 bzw. 4 Byte pro Parameter benötigen. Zur Interferenz können optional diese Parameterformate durch Quantifizierung weiter verringert werden auf FP8 oder FP4 mit 1 Byte or 0,5 Byte pro Parameter. Zusätzlich muss das Modell während des Trainings circa 4 mal in den Speicher geladen werden, um die Backpropagation durchzuführen. Zur Inferenz der Modelle ist dementsprechend ein Speicherbedarf von $\frac{1}{4}$ der Trainingswerte bei Beibehaltung des Parameterformates notwendig. Modelle, die sehr populär geworden sind, aber nicht zur

Auswahl stehen, sind BERT, RoBERTa, DistillBERT, BART, T5, Opus, Pegasus, DialoGPT, Blenderbot und Flan-T5.

Wenn die Aufgabe des Question Answering gelöst werden soll, ist ein BERT-basiertes Modell die erste Wahl. Modelle wie BERT, RoBERTa und DistillBERT besitzen aufgrund ihrer Encoder-Architektur optimale Eigenschaften für die Informationsextraktion aus Text. Dieser Ansatz ist jedoch in zwei Punkten eingeschränkt. Die Modelle lernen nicht Fakten und können auf Basis einer Frage neue Antworten formulieren, sondern finden Textstellen in einem Kontext, die die Frage beantworten. Es findet keine Textgenerierung statt, sondern es werden Ausschnitte aus einem gegebenen Textkontext als Antwort gefunden. Dieser Kontext ist wiederum durch die Größe des Modells begrenzt und beschränkt sich in den meisten Fällen auf 512 Tokens. Ist die Antwort auf die Frage nicht im Kontext enthalten, kann das Modell keine Antwort finden. Da die Fragen unabhängig vom Kontext beantwortet werden sollen, können diese Modelle nicht verwendet werden. Andere Modelle wie BART, T5, Opus, Pegasus, DialoGPT, Blenderbot und Flan-T5 wurden für andere Aufgaben entwickelt und können nicht auf die vorliegende Aufgabe angewendet werden. Sie sind für Textklassifikation, Textübersetzung, Textzusammenfassung, Dialoge oder Text-2-Text-Aufgaben konzipiert.

GPT-2 ist das kleinste Modell der OpenAI GPT-Serie und wurde erstmals in Radford u. a. (2019) vorgestellt. Aufgrund seiner geringen Größe kann das Modell auf einzelnen Graphics Processing Unit (GPU)s trainiert und verwendet (inferiert) werden, was die Nutzung des Modells erheblich vereinfacht. Außerdem ist es frei verfügbar und kann ohne weitere Maßnahmen verwendet werden. Allerdings ist die Performance in der Einstellung ZeroShot nicht ausreichend. Das Modell erreicht eine Imitation von Texten, aber kein Verständnis und keine Wiedergabe von Fakten.

GPT-3 und das darauf basierende GPT-3.5 (ChatGPT), vorgestellt in Brown u. a. (2020), ist zum Zeitpunkt der Arbeit frei nutzbar, hat eine sehr gute ZeroShot-Performance und ist groß genug, um als QA-Modell zu dienen. Das Modell ist jedoch nicht frei verfügbar. Es existiert zwar eine Application Programming Interface (API), diese dient jedoch nur zur Inferenz des Modells, ein Training ist nicht möglich. Damit scheidet das Modell bereits aus. Ein weiteres Problem ist die Größe des Modells. Bei ca. 175 Milliarden Parametern ist eine Nutzung des Modells erst mit 8 A100 GPUs möglich (eine Nvidia A100 GPU besitzt 80 GB VRAM). Das Training benötigt zusätzlich etwa den 4-fachen Speicher. Dieser Speicher kann zwar vom Rechenzentrum der Universität Leipzig zur Verfügung gestellt werden, verhindert aber

den Einsatz des Modells in einer längerfristigen Umgebung.

GPT-4 ist das mächtigste Modell von OpenAI und wurde in OpenAI (2023) vorgestellt. Auch dieses Modell ist nicht frei verfügbar und kann nur über eine API verwendet werden. Die Leistungsfähigkeit von GPT-4 ist jedoch deutlich höher als die von GPT-3, weshalb die Untersuchung dieses Modells als Ersatz für ein speziell trainiertes Modell geplant ist. Die Größe des Modells wurde von OpenAI nicht veröffentlicht, ist aber definitiv größer als GPT-3, weshalb eine Verwendung hier allein aufgrund der Größe ausscheidet.

GPT-J erreicht mit 6,7 Milliarden Parametern die Größe der kleinsten Modellversion der GPT-3 Serie, zeigt aber eine deutlich bessere Performance als GPT-2. Das Modell ist unter Wang und Komatsuzaki (2021) veröffentlicht und frei verfügbar. Die Leistungsfähigkeit des Modells ist im Vergleich zu den anderen Modellen in der Auswahl geringer und wird daher ausgeschlossen.

GPT-NeoX wurde in Black u. a. (2022) vorgestellt und erreicht mit einer Größe von 20 Milliarden Parametern die Leistungsfähigkeit von GPT-3 mit 175 Milliarden Parametern. Das Modell ist frei verfügbar und kann ohne weitere Maßnahmen verwendet werden. Die Leistung des Modells ist vergleichbar mit GPT-3, während die Größe des Modells ein Training mit 8 A100 GPUs erlaubt. Diese Leistung wird jedoch von LLaMA übertroffen und daher nicht ausgewählt.

Die in Touvron u. a. (2023a) vorgestellten LLaMA-Modelle haben einen besonderen Ansatz. Durch den Fokus auf längeres Training und größere Datensätze bieten sie parametereffiziente Modelle, die sehr gute Leistungen erzielen können. Die Modelle von LLaMA werden ausgewählt, um die bestmögliche Performance mit den kleinstmöglichen Modellen zu erreichen. Hier stellt sich die Frage, welches Modell aus der LLaMA-Reihe verwendet werden soll. Die Modelle unterscheiden sich sowohl in der Größe als auch proportional in der Leistung.

Die Tabelle 4.2 zeigt die Leistungsfähigkeit der Modelle im Vergleich zu anderen Modellen und bei verschiedenen Einstellungen (ZeroShot bis 64Shot). Selbst das kleinste Modell mit 7 Milliarden Parametern übertrifft die Leistung von GPT-3 und kann während der Inferenz auf einer einzelnen V100 GPU verwendet werden (eine Nvidia V100 GPU besitzt 32 GB VRAM). Nach Angaben der Autoren kann auch das Modell mit 13 Milliarden Parametern auf einer einzigen V100 GPU eingesetzt werden, was für die Wahl dieses Modells spricht. Das 33-Milliarden-Parameter-Modell erzielt im ZeroShot in Teilen sogar etwas bessere Ergebnisse als das 65-Milliarden-Parameter-Modell, weshalb eine Wahl des 33 Milliarden oder 65 Milliarden Parameter Modells

		BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA
GPT-3	175B	60,5	81,0	-	78,9	70,2	68,8	51,4	57,6
Gopher	280B	79,3	81,8	50,6	79,2	70,1	-	-	-
Chinchilla	70B	83,7	81,8	51,3	80,8	74,9	-	-	-
PaLM	62B	84,8	80,5	-	79,7	77,0	75,2	52,5	50,4
PaLM-cont	62B	83,9	81,4	-	80,6	77,0	-	-	-
PaLM	540B	**88,0**	82,3	-	83,4	**81,1**	76,6	53,0	53,4
LLaMA	7B	76,5	79,8	48,9	76,1	70,1	72,8	47,6	57,2
	13B	78,1	80,1	50,4	79,2	73,0	74,8	52,7	56,4
	33B	83,1	82,3	50,4	82,8	76,0	**80,0**	**57,8**	58,6
	65B	85,3	**82,8**	**52,3**	**84,2**	77,0	78,9	56,0	**60,2**

Tabelle 4.2: Zero-Shot Leistung verschiedener Modelle in Begründungsaufgaben für gesunden Menschenverstand. Veröffentlicht in Touvron u. a. (2023a)

hier sinnvoll ist. Mit 33 Milliarden Parametern und der Verwendung von FP16 Werten (auch Half-Precision genannt) benötigt das Modell 66 GB GPU RAM, um für die Inferenz verwendet werden zu können. Für das Training werden 264 GB GPU RAM benötigt, was 4 A100 GPUs entspricht. Das Modell mit 65 Milliarden Parametern benötigt 7 A100 GPUs für das Training und 2 A100 GPUs für den Betrieb. Die Modelle sind jedoch nicht öffentlich verfügbar, sondern werden nur zu Forschungszwecken über einen Antrag weitergegeben. Dieser Antrag wurde im Rahmen dieser Arbeit gestellt und genehmigt.

Eine Alternative zu den LLaMA-Modellen ist OpenLLaMA, veröffentlicht unter Geng und Liu (2023). Die hier verfügbaren Modelle mit 3 Milliarden, 7 Milliarden und 13 Milliarden Parametern entsprechen in ihrer Leistungsfähigkeit den LLaMA-Modellen, sind aber unter der Open-Source-Lizenz Apache 2.0 verfügbar. Da hier jedoch keine Publikation über den Prozess der Erstellung dieser Modelle vorhanden ist, wird von der Verwendung dieser Modelle vorerst abgesehen.

Zuletzt veröffentliche MetaAI die Llama 2 Modelle, welche auf der Architektur und Trainingsart von den LLaMA Modellen basieren, jedoch in allen Bereichen eine bessere Leistung durch mehr Training und größere Datensätze erzielen. Die Llama 2 Modelle sind unter Touvron u. a. (2023b) veröffentlicht und stehen ebenso unter eine Lizenz, die nur zu Forschungszwecken freigegeben wird. Llama 2 ist in den Größen 7 Milliarden, 13 Milliarden und 70 Milliarden Parametern verfügbar.

In dieser Arbeit wird zunächst das Modell Llama 2 7B ausgewählt, da es eine sehr gute Leistung mit minimalen Ressourcenanforderungen bietet und bessere Leistungen als die Vorgänger Version der LLaMA Modelle erzielt. Eine Auswahl dieser Modellgröße begrün-

det sich daher, dass eine Nutzung der Modelle über 13 Milliarden Parameter auf einzelnen GPUs nicht möglich ist, und dadurch die Nutzbarkeit eingeschränkt wird. Ebenso übersteigen die technischen Speicheranforderungen größerer Modelle die Kapazitäten einzelner Knoten im genutzten GPU-Computing Cluster, weshalb dadurch die technische Umsetzung des Trainings drastisch in ihrer Komplexität zunimmt. Einzelne Tests der Knoten-übergreifenden Kommunikation zeigten grundlegende Probleme und sollten deshalb vermieden werden. Auf weitere Einzelheiten über den Aufbau der verwendeten GPU-Computing Clusters wird in Abschnitt 4.3.1 eingegangen.

4.2 DATENKURATION

Um die Literatur optimal nutzen zu können, muss eine Datenkuration durchgeführt werden. Die Daten liegen im Epub-Format vor, können aber in diesem Format nicht verarbeitet werden, da das Modell nur Text verarbeiten kann. Die Datenkuration umfasst mehrere Schritte, die im Folgenden beschrieben werden.

4.2.1 *Extraktion des Textes*

Die Extraktion des Textes wird händisch durchgeführt. Dazu wird das Epub-Dokument geöffnet und der Text kopiert. Der Text wird in einem Texteditor eingefügt und als `.md` Datei gespeichert. `.md`-Dateien folgen dem Markdown-Format[1] und haben die besondere Eigenschaft, mit Hilfe von Text-Symbolen Formatierungen wie Überschriften, Aufzählungen und Tabellen darzustellen.

4.2.2 *Unverständliche Formate*

Während der Extraktion des Textes fallen alle Bilder als auch Kopf- und Fußzeilen weg. Da eine einzelne Text-Datei keine sinnvolle Unterteilung in Seiten hat, sind Kopf- und Fußzeilen und damit einhergehend Seitenzahlen nicht relevant. Bilder können nicht von dem Modell verarbeitet werden und werden daher entfernt. Die Bildunterschriften als auch Referenzen auf Bilder stehen somit ohne Kontext in den Kapiteln und werden ebenso entfernt.

4.2.3 *Textpassagen ohne Wissen oder Kontext*

Teile des Dokuments enthalten kein Wissen, stehen in keinem Zusammenhang oder sind für das Modell unverständlich formatiert. Die

1 `https://markdown.de/` (abgerufen am 28.6.2023)

Titelseite wird auf den Buchtitel reduziert, das Vorwort wird weggelassen, da es kein Wissen zum Thema enthält, Tabellen- und Abbildungsverzeichnisse werden ebenfalls entfernt. Bild- und Tabellenverzeichnisse enthalten zwar Wissen, können aber nicht in Kontext gesetzt werden, da die verwendeten Aufmerksamkeitsmechanismen des Modells diese Verzeichnisse nicht mit Texten in späteren Kapiteln verknüpfen können, da der Kontext zu groß ist. Auch die Vorstellung der Autoren enthält kein Wissen zum Thema und wird daher entfernt.

4.2.4 *Optionale Textentfernungen*

Fortführend kann geprüft werden, ob das Weglassen von Literaturverweisen, Überschriften und Tabellen zu einer besseren Leistung führt. Literaturverzeichnisse beziehen sich auf das vorhergehende Kapitel, haben aber ein schwieriges Format. Insbesondere Uniform Resource Locator (URL)s müssen vom Modell in Tokens zerlegt werden, die nicht den Inhalt der URLs widerspiegeln. Überschriften haben im Vergleich zum Text nur einen minimalen Anteil an der Gesamtzahl der Tokens, wodurch das in ihnen enthaltene Wissen verloren geht. Die menschenlesbare Formatierung einer Überschrift hat keinen Einfluss auf die Bewertung durch das Modell. Tabellen liegen in einem Format vor, das vom Modell nicht nativ verarbeitet werden kann. Selbst die Tabellendarstellung in Markdown ist zwar einlesbar, aber es ist fraglich, ob sie verständlich ist. Aufgrund der geringen Größe des Datensatzes ist auch nicht zu erwarten, dass dieses Format erlernt wird. Daher wird hier eine Aufzählung als Tabellenersatz verwendet.

4.2.5 *Bleibende Texte*

Das Abkürzungsverzeichnis und das Glossar bleiben im Text enthalten, da sie für das Verständnis der Fragen wichtige Begriffe definieren. Einige Fragen können Akronyme oder Fachbegriffe verwenden, die vom Modell verstanden werden müssen.

4.2.6 *Formatierung von Text*

Die Formatierung des Textes im Epub-Dokument kann nicht berücksichtigt werden. Daher muss hier ein Ersatz in Form einer Markdown-Formatierung gefunden werden. Überschriften werden durch „#“, Aufzählungen durch „-“ und „1.“ dargestellt. Tabellen werden durch eine Aufzählung dargestellt, wobei die erste Zeile die Überschriften enthält und die zweite Zeile die Trennung zwischen Überschriften und Inhalt darstellt. Die Trennung zwischen den Spalten erfolgt durch „|“. Hoch- und tiefgestellte Zahlen werden durch „^“ und „_“ dargestellt. Formeln werden in den LaTeX Mathe-Modus umgewandelt.

4.2.7 *Potentielle Extraktion von Fragen*

Neben der Verwendung des Textes als Eingabe in das Modell ist auch eine weitergehende Nutzung möglich. Die Übungsabschnitte sowie das Glossar und das Abkürzungsverzeichnis können als Quelle für Fragen genutzt werden, um das Modell und andere Modelle besser bewerten zu können.

4.3 UNÜBERWACHTES WEITERTRAINIEREN

Um das Llama-Modell zu trainieren, wird ein Programm benötigt, das den Datensatz in Tokens umwandelt und diese dann in Batches an das Modell übergibt. Basierend auf diesen Batches wird das Modell verwendet, um die Gewichte mittels Backpropagation anzupassen. Dieser Prozess stellt somit eine Epoche dar. Bei kleineren Datensätzen und Finetuning werden oft mehrere Epochen gewählt, während größere Datensätze (mehrere hundert Gigabyte) nur eine Epoche oder nicht einmal eine Epoche benötigen. Mit zunehmender Epoche passt sich das Modell immer mehr an die Literatur an, bis eine Grenze überschritten wird. Diese Grenze wird oft als „Overfitting“ bezeichnet und stellt ein Modell dar, das sich zu sehr an einen Datensatz angepasst hat. Dadurch verliert das Modell die Fähigkeit, auf unbekannte Daten zu verallgemeinern, was in diesem Anwendungsfall dazu führt, dass keine Fragen mehr beantwortet werden können.

Um Overfitting zu vermeiden, wird der Datensatz in einen Trainings- und einen Validierungsdatensatz aufgeteilt. Dabei wird eine Aufteilung von 95 % Training und 5 % Validierung verwendet. Das Modell wird nur auf dem Trainingsdatensatz trainiert, aber in regelmäßigen Abständen mit dem Validierungsdatensatz überprüft. Die Ergebnisse des Validierungsdatensatzes haben keinen Einfluss auf die Anpassung der Gewichte, sondern dienen nur zur Überprüfung der Generalisierungsfähigkeit des Modells. Sowohl für den Trainings- als auch für den Validierungsdatensatz wird ein Absinken der Fehlerfunktion erwartet. Dieses Absinken ist ein Indikator dafür, dass das Modell sich an den Datensatz anpasst. Sobald das Absinken auf dem Validierungsdatensatz aufhört, ist das Training abgeschlossen. Die Ergebnisse zeigen in den meisten Fällen, dass die Fehlerfunktion des Trainingsdatensatzes niedriger als die des Validierungsdatensatzes ist. Solange der Unterschied zwischen den beiden Fehlerfunktion nicht zu groß ist, konnte das Modell erfolgreich trainiert werden.

Für die Umsetzung dieser Trainingsschritte wird die Programmiersprache Python verwendet, da sie eine Vielzahl von Bibliotheken für die Textverarbeitung und den Einsatz von neuronalen Netzen bietet. Die Bibliothek Transformers von HuggingFace (Wolf u. a., 2020) bietet

Parameter	Wert
Lernrate	$3e^{-4}$
AdamW β_1	0,9
AdamW β_2	0,95
Gewichtsreduktion	0,1
Gradientenlimitierung	1,0
Aufwärmphase	0
Epochen	3

Tabelle 4.3: Parameter für das Training des Llama 2 7B-Modells

eine Vielzahl von Modellen und Trainingsmethoden und basiert auf der populären Bibliothek PyTorch (Paszke u. a., 2019). PyTorch bietet mit Hilfe von CUDA die Möglichkeit, Trainingsschritte auf einer Nvidia GPU durchzuführen, was die in Abschnitt 2.2.3 beschriebenen Schritte um ein Vielfaches beschleunigt. Mit Hilfe von DeepSpeed (Rajbhandari u. a., 2020) kann dieses Training auch auf mehrere GPUs verteilt werden. Dies ist notwendig, da das verwendete Modell mit 7 Milliarden Parametern zu groß ist, um in den Arbeitsspeicher einer einzelnen GPU zu passen.

Vor Beginn des Trainings müssen einige Konfigurationsparameter festgelegt werden. Diese Parameter sind in der Tabelle 4.3 aufgelistet und folgen den Trainingsparametern von Touvron u. a. (2023a).

Die Lernrate (engl. „Learning Rate“) ist ein Parameter, der das Ausmaß der Anpassung der Gewichte bestimmt (siehe Definition 26). Sie wird während des Trainings angepasst, um eine schnelle Konvergenz zu ermöglichen. Diese Anpassung folgt dem Kosinus-Lernratenschema, so dass die endgültige Lernrate 10 % des Anfangswertes beträgt. Die Lernraten hängen von der Modellgröße ab. Im Allgemeinen benötigen größere Modelle eine niedrigere Lernrate. Der Optimierer AdamW berücksichtigt den zeitlichen Abfall der Gewichte und wurde in Loshchilov und Hutter (2019) vorgestellt. Er ist die häufigste Wahl in aktuellen Modellen und wurde auch für das Training der Llama-Modelle verwendet.

Gewichtsabnahme (engl. „Weight Decay“) ist ein Parameter, der die Gewichte während des Trainings reduziert. Er stellt eine Regularisierung dar, die Overfitting verhindern soll. 0,1 entspricht einer Gewichtsreduktion von 10 %.

Die Gradientenbegrenzung (engl. „Gradient Clipping“) ist ein Parameter, der die Größe der Gradienten begrenzt. Dies ist notwendig, da

bei großen Modellen die Gradienten während der Backpropagation mit zunehmender Tiefe immer größer werden. Ohne eine Begrenzung würden die Gradienten hier exponentiell wachsen. Eine Gradientenbegrenzung beschränkt die Gradienten auf 1,0.

Die Aufwärmphase beschreibt die Anzahl der Schritte, die das Modell benötigt, um die Lernrate von 0 linear auf den Anfangswert zu erhöhen. Dies ist notwendig, da sich die Modelle zu Beginn des Trainings zu schnell an nicht universelle Muster anpassen. Normalerweise wird dies durch ein längeres Training korrigiert, bei dem die zuerst gelernten Muster wieder verlernt werden müssen. Durch die Aufwärmphase wird ein längeres Training vermieden. Da in diesem Fall das Modell bereits auf einem großen Datensatz trainiert wurde, ist eine Aufwärmphase nicht notwendig.

Der Artikel von Touvron u. a. (2023a) beschreibt ein Training über eine Epoche. Da in diesem Fall jedoch der Datensatz im Vergleich minimal ist, wird eine größere Anzahl von Epochen verwendet.

4.3.1 *Ausführen der Training-Programme*

Zur Ausführung der Trainingsprogramme wird SLURM[2] verwendet. Mit Hilfe von SLURM-Skripten können Programme über mehrere Knoten, Central Processing Unit (CPU)s und GPUs verteilt berechnet werden. Diese Skriptsprache wird vom Rechenzentrum der Universität Leipzig verwendet, um verschiedene Programme auf den Supercomputern Clara und Paula laufen zu lassen.

Die Supercomputer Clara und Paula sind Rechennetze, die auf GPU beschleunigte Berechnungen spezialisiert sind. Paula besteht aus 12 Knoten mit je 8 Nvidia Tesla A30 GPUs. Clara ist zweigeteilt und besteht aus 8 Knoten mit je 4 Nvidia Tesla V100 GPUs und 22 Knoten mit je 8 Nvidia GeForce RTX 2080 TI GPUs. Die Knoten sind über ein Infiniband-Netzwerk verbunden, das eine hohe Bandbreite und geringe Latenz bietet.

Supercomputer sind gemeinsam genutzte Ressourcen, die von mehreren Nutzern gebucht werden können. Die Auswahl der Supercomputer hängt also davon ab, welche Knoten gerade frei sind. Generell werden Clara-Nodes mit V100 GPUs bevorzugt, da hier der größtmögliche Speicher pro GPU zur Verfügung steht. Dies hat zur Folge, dass weniger GPUs gleichzeitig reserviert werden müssen, was die Wartezeit verkürzt und die Kommunikation zwischen den GPUs minimiert.

2 `https://slurm.schedmd.com/overview.html` (abgerufen am 16.6.2023)

Die Trainingsprogramme müssen auch in der Lage sein, mit der Multi-GPU-Architektur umzugehen. Dazu wird die Bibliothek DeepSpeed (Rajbhandari u. a., 2020) in Kombination mit der Bibliothek Transformers verwendet. DeepSpeed erlaubt verschiedene Arten der Parallelisierung, der Fokus liegt hier auf Zero Redundancy Optimizer (ZeRO) 2 (Rajbhandari u. a., 2020). Die ZeRO Parallelisierung erlaubt eine verteilte parallele Datenverarbeitung, bei der das Modell auf mehrere GPUs aufgeteilt wird. Ohne diese Option ist ein Training nicht möglich, da, wie oben beschrieben, einzelne Modelle nicht vollständig auf eine GPU passen. Genauere Erläuterungen zur Konfiguration von DeepSpeed und zur Verwendung der Transformer-Bibliothek sind in Kapitel 5 beschrieben.

4.4 KLAUSURFRAGEN

Zur Überprüfung der Modelle werden Klausurfragen verwendet. Diese Klausurfragen ergeben sich aus den mündlichen Prüfungsfragen des Moduls „Architektur von Informationssystemen im Gesundheitswesen“, den schriftlichen Prüfungsfragen des Moduls „Informationssysteme in medizinischer Versorgung und Forschung“ und aus Teilen des Buches Winter u. a. (2023). Die Fragen sind in drei Kategorien unterteilt:

1. Einzelfragen, die einen bestimmten Sachverhalt abfragen
2. Multi-Fakten-Fragen, die mehrere Fakten abfragen
3. Transferfragen, die einen Sachverhalt von einem Kontext in einen anderen übertragen

Allerdings ist die Anzahl der durchgeführten Klausuren zum Zeitpunkt der Erstellung dieser Arbeit geringer als erwartet, da der Studiengang erst seit 2021 existiert. Dies führt zu einer geringeren Qualität der Bewertung, hat aber keinen Einfluss auf die Qualität des Modells. Beispiele für die drei Kategorien sind in Tabelle 4.4 zu finden.

Die Llama-Modelle sind ohne weiteres Finetuning verfügbar und wurden daher nicht an die Aufgabe der Beantwortung von Fragen angepasst. Dies hat zur Folge, dass die Modelle bei einer Frage wie „Welche Farbe hat der Himmel?“ nicht die Antwort liefern, sondern den Text im gleichen Stil mit weiteren Fragen fortsetzen. Um eine Antwort auf eine Frage zu erhalten, muss die Frage in eine unvollständige Aussage umformuliert werden. Im gegebenen Beispiel wird aus „Welche Farbe hat der Himmel?“ die Aussage „Der Himmel hat die Farbe“, die dann vom Modell mit „blau.“ beantwortet wird. Dieser Prozess der Umformulierung von Fragen wird für alle vorhandenen Fragen durchgeführt. Das GPT-4-Modell wurde auf die Aufgabe der

Kategorie	Frage	Antwort
Einzelfrage	Define the term „tHIS“	tHIS stands for transinstitutional health information system, an information system of a health care network which consists of multiple health care settings (contexts). It comprises all data, information, and knowledge processing as well as the associated human or technical actors in their respective data, information, and knowledge processing roles.
Multi-Fakten Frage	Compare HL7 V2 and HL7 FHIR. Explain at least three differences	Both used to exchange health information between application systems Syntax HL7 v2: proprietay, ASCII-text Syntax HL7 FHIR: XML / JSON = easier to implement Structure HL7 v2: fixed, hierarchical, fixed amount of segments and codes Structure HL7 FHIR: flexible, modular, resource/element based
Transferfrage	When is a model good in the context of health information systems?	A good models should be able to help understand and predict the behavior of the system or process. It should also be able to help design and evaluate health information systems. A reference architecture can be used to support the design of a proper HIS architecture that meets the various stakeholder concerns of HISs. This architecture should be able to show the HIS from a different angle, suitable for various stakeholders.

Tabelle 4.4: Beispielfragen der drei Fragekategorien

Beantwortung von Fragen trainiert und kann daher mit normalen Fragen evaluiert werden.

Einige Fragen erfordern das Zeichnen von Systemen oder das Verbinden von Begriffen mit Pfeilen. Da das Modell keine Bilder erzeugen kann, werden diese Fragen so umformuliert, dass sie als Textfragen beantwortet werden können. Eine Skizze wird entsprechend in eine Systembeschreibung umformuliert und eine Begriffsverbindung in eine Begriffsliste. Andere Fragen bauen auf Lösungen der vorhergehenden Frage auf. Dies ist nur dann sinnvoll, wenn das Modell den Verlauf der Fragestellungen kennt, was nicht der Fall ist. Diese Fragen müssen ebenfalls umformuliert werden, um unabhängig von anderen Fragen zu sein.

4.5 MODELLEVALUATION

Der Vergleich der Modelle erfolgt im Rahmen einer manuellen Bewertung. Dazu werden die in Abschnitt 4.4 definierten Fragen verwendet. Diese Fragen werden von dem Llama 2 7B Modell sowohl vor als auch nach dem Continual Pretraining und von GPT-4 beantwortet. Die Antworten werden manuell ausgewertet und mit den Antworten aus dem Buch Winter u. a. (2023) verglichen. Die Lösungen werden ebenfalls in drei Kategorien eingeteilt:

1. Richtig, wenn die Antwort mit der Antwort im Buch übereinstimmt
2. Falsch, wenn die Antwort falsches Wissen enthält
3. Nicht beantwortet, wenn der generierte Text keinen Bezug zur Frage hat

Die Modelle werden mit Hilfe der Makro-F1 Bewertung verglichen. Die Definition folgt den unter Omar u. a. (2023) beschriebenen Formeln:

$$C = S \cap G \tag{4.2}$$

$$prec = \frac{|C_q|}{|S_q|} \tag{4.3}$$

$$recall = \frac{|C_q|}{|G_q|} \tag{4.4}$$

$$F1 = 2\frac{prec \cdot recall}{prec + recall} \tag{4.5}$$

Dabei ist S_q die Menge der für die Frage q generierten Antworten, G_q die Menge der richtigen Antworten aus dem Buch Winter u. a. (2023) und C_q die Menge der richtigen Antworten, die sowohl in S_q als auch in G_q enthalten sind. *prec* ist die Präzision, *recall* der Recall und $F1$ der F1-Wert pro Frage.

Definition 34 ***Präzision*** *ist der Anteil der richtigen Antworten an allen Antworten des Modells.*

Definition 35 ***Recall*** *ist der Anteil der richtig Antworten an alle korrekten Antworten des Evaluierungsdatensatzes.*

Definition 36 *Der* ***F1*** *Wert ist das harmonische Mittel aus Präzision und Recall.*

Definition 37 *Der* ***Mikro-F1*** *Wert beschreibt einen Gesamt-F1 Wert über alle Fragen und deren Antworten.*

Definition 38 *Der* ***Makro-F1*** *Wert beschreibt das Mittel aller F1 Werte für jede Frage. In einem unbalancierten Datensatz ist dies eine bessere Metrik, um alle Fragen gleich zu gewichten.*

Fragen gelten als beantwortet, wenn sie den Kategorien 1 und 2 entsprechen. Dieselben Berechnungen werden insgesamt und getrennt für alle drei Fragekategorien durchgeführt. Es wird der Makro-F1 Wert genutzt um eine bessere Metrik unabhängig einzelner Ausreißer zu erhalten. Wie in Usbeck u. a. (2019) beschrieben, werden größtenteils Makro-F1 Werte in Evaluierung verwendet. Der Mikro-F1 Wert ist sensitiv zu einzelnen Fehlschlägen des Modells. Generiert z.B. das Modell 1000 falsche Antworten zu einer aus 100 Fragen, beantwortet jedoch alle anderen Fragen korrekt, sinkt der Mikro-F1 Wert stark. Jedoch ist das Modell in diesem Fall als sehr gut anzusehen, da es 99% der Fragen korrekt beantwortet. Der Makro-F1 Wert ist hier repräsentiver, da er jede einzelne Frage gleich gewichtet.

Den in Omar u. a. (2023) beschriebenen Ansatz folgend werden die Modelle auf die Kriterien Korrektheit, Determinismus, Robustheit, Erklärbarkeit und Fragenverständnis evaluiert. Die Kriterien „Nutzung aktueller Informationen“ und „Generalisierung über verschiedene Domänen“ werden nicht betrachtet, da die Modelle keinen Zugriff auf aktuelle Informationen und verschiedene Domänen haben.

Korrektheit:
Die Korrektheit wird durch die oben beschriebenen Formeln berechnet und verglichen. Eine höhere Korrektheit entspricht einem besseren Ergebnis. Bei der Anwendung des Modells ist eine hohe Präzision wichtiger als ein hoher Recall, da eine falsche Antwort für einen unkundigen Anwender wesentlich schädlicher ist als keine Antwort. Die Modelle sollen immer keine Antwort einer falschen Antwort vorziehen.

Determinismus:
Determinismus beschreibt die Reproduzierbarkeit von Antworten. Für die gleiche Frage und das gleiche Modell sollen die Antworten immer

gleich sein. Sprachmodelle zeigen ein gewisses Maß an nichtdeterministischem Verhalten, daher ist diese Statistik ein wichtiges Kriterium. Um dies zu erreichen, werden Fragen 3 mal an die Modelle gestellt und die beste Antwort wird zur Berechnung der Korrektheit verwendet. Dabei wird auch diese Statistik berechnet. Eine Antwort gilt als deterministisch, wenn die gleichen Fakten mit leichten Umformulierungen in der Antwort enthalten sind.

Robustheit:
Die Robustheit beschreibt die Fähigkeit des Modells, auch bei fehlerhaften Eingaben korrekte Antworten zu generieren. Dazu werden grammatikalische Fehler sowie Rechtschreibfehler in eine Teilmenge der Fragen eingefügt und die Korrektheit der Antworten berechnet. Eine Antwort wird als robust bezeichnet, wenn sie auch bei fehlerhaften Eingaben korrekt ist.

Erklärbarkeit:
Die Erklärbarkeit beschreibt die Fähigkeit des Modells, die erzeugten Antworten zu erklären. Dazu werden die Modellantworten manuell daraufhin überprüft, ob sie eine Erklärung enthalten. Eine Antwort gilt als erklärbar, wenn sie eine Erklärung enthält.

Fragenverständnis:
Das Fragenverständnis beschreibt die Fähigkeit des Modells, die Frage zu verstehen. Dazu werden die Antworten der Modelle manuell daraufhin überprüft, ob sie die Frage verstanden haben. Eine Frage gilt als verstanden, wenn sich die Antwort inhaltlich auf die Frage und deren Sachverhalt bezieht. Die Antwort muss nicht korrekt sein.

5 AUSFÜHRUNG DER LÖSUNG

Die Lösung gliedert sich in sechs Schritte, die für die Durchführung des Trainings und die Bewertung der Ergebnisse notwendig sind. Im Folgenden werden diese Schritte näher erläutert. Für jeden Schritt sind die verwendeten Techniken und Bibliotheken separat aufgeführt. Die Schritte sind wie folgt gegliedert:

1. Laden des Modells
2. Modelltraining
3. Generierung von Antworten auf dem Evaluierungsdatensatz
4. Bewertung der erzeugten Antworten
5. Auswertung basierend auf den Bewertungen

5.1 HERUNTERLADEN DES MODELLS

Die Llama-Modelle werden unter einer nicht-kommerziellen Lizenz für Forschungszwecke zur Verfügung gestellt. Der Zugang zu den Modellen wird im Einzelfall auf Anfrage gewährt. Diese Anfrage wurde im Rahmen dieser Arbeit gestellt und bestätigt. Anschließend kann ein von den Autoren vorbereitetes Skript oder die API von Huggingface mit zugehörigem Authentifizierungs-Token verwendet werden, um die trainierten Gewichte des Modells herunterzuladen. Das Skript benötigt eine explizite URL, die nach einer vorgegebenen Zeit von einer Woche nach Freigabe der Modelle ungültig wird. Aus diesem Grund ist diese URL nicht im Skript enthalten.

5.2 TRAINING DES MODELLS

Um das Llama-Modell zu trainieren, wird die Transformers Bibliothek von Huggingface verwendet (Wolf u. a., 2020). Das in Python geschriebene Trainingsskript basiert auf dem Beispielskript zum Training von kausalen Sprachmodellen aus dem Huggingface GitHub Repository[1] und wurde teilweise an die Anforderungen des hier durchgeführten Trainings angepasst. Die Konfiguration des Trainings gliedert sich in vier Bereiche: Einstellungen für das Modell, Einstellungen für die Trainingsdaten, Einstellungen für das Training selbst und Einstellungen für DeepSpeed. Im Folgenden werden diese Einstellungen näher erläutert.

1 `https://github.com/huggingface/transformers/blob/v4.31.0/examples/pytorch/language-modeling/run_clm.py` abgerufen am 16.8.2023

Parameter	Wert
model_name	meta-llama/Llama-2-7b-hf
cache_dir	./cache
use_fast_tokenizer	false
model_revision	main
use_auth_token	true
hugging_token	*Huggingface Token*
torch_dtype	auto
low_cpu_mem_usage	false

Tabelle 5.1: Parameter zur Auswahl und Konfiguration des Modells

5.2.1 *Konfiguration des Modells*

Die zur Auswahl und Konfiguration des Modells verwendeten Parameter sind in Tabelle 5.1 aufgelistet.

Der Parameter `model_name` entspricht einer Modellauswahl und kann entweder ein relativer Pfad zu einem lokalen Modell oder ein Modellname sein. Der Modellname wurde hier auf das Modell Llama 2 7B gesetzt und verweist auf das von Huggingface gehostete Modell in einem mit der Transformers-Bibliothek kompatiblen Format. Llama 2 wurde im Juli 2023 von Meta AI veröffentlicht und stellt eine generelle Verbesserung der ursprünglichen Llama 1 Modelle dar (Touvron u. a., 2023b). Neben der nun auf 4096 Tokens erweiterten Kontextlänge (die ursprüngliche Kontextlänge von Llama 1 lag bei 2048 Tokens) stellt Meta AI Llama 2 auch in einer Chat-Version zur Verfügung. Die Chat-Version wurde zusätzlich mit Hilfe von Reinforcement Learning aus menschlichem Feedback trainiert und ermöglicht so eine einfachere Nutzung der vortrainierten Modelle im Kontext eines Chatbots. Diese Chat-Modelle wurden jedoch nicht zum Training verwendet, da davon ausgegangen wird, dass dieses zusätzliche Training durch das hier durchgeführte Continual Pretraining überschrieben wird. Die Llama 2 Modelle wurden unter anderem im Artikel von Touvron u. a. (2023b) vorgestellt.

Der Parameter `cache_dir` beschreibt den Speicherort des heruntergeladenen Modells. Dies ermöglicht ein wiederholtes Trainieren des Modells ohne erneuten Download. Die heruntergeladenen Modelle werden durch das Training nicht überschrieben und stellen somit den Grundzustand des Modells dar.

`use_fast_tokenizer` konfiguriert die Verwendung einer schnelleren Version des Tokenizers zur Konvertierung der Datensätze. Diese Option ist optional und wurde hier nicht verwendet.

`model_revision` beschreibt die zu verwendende Version des Modells. Hier wurde die aktuellste Version verwendet, die durch den Wert „main“ repräsentiert wird.

`use_auth_token` und `hugging_token` beschreiben die Verwendung eines Authentifizierungs-Tokens, um Modelle von Huggingface herunterzuladen. Diese Authentifizierung ist notwendig, da auch die Llama 2 Modelle unter der gleichen Lizenz wie Llama 1 stehen und nur auf Anfrage zur Verfügung gestellt werden. Um die Modelle mit beschränktem Zugang herunterzuladen, wurde ein Huggingface-Account erstellt, die Anfrage an Meta AI zur Nutzung der Llama 2 Modelle gestellt und bestätigt und ein Authentifizierungstoken generiert.

Der Parameter `torch_dtype` beschreibt den Datentyp, der für die Darstellung der Modellparameter verwendet wird. Hier stehen Float32, Float16 und BFloat16 zur Verfügung. Vorgefertigte Modelle wurden ursprünglich mit einem Datentyp erstellt und müssen in einen anderen Datentyp konvertiert werden, wenn der `torch_dtype` nicht übereinstimmt. Diese Konvertierung ist rechenintensiv und kann dazu führen, dass Modelle fehlerhaft trainiert werden oder mehr Rechenleistung während der Inferenz und des Trainings benötigen. Aus diesem Grund wurde hier der Parameter „auto“ verwendet, der den Datentyp des Modells automatisch erkennt und benutzt.

`low_cpu_mem_usage` beschreibt ein Verfahren der Bibliothek Transformers zum Laden großer Modelle auf Systemen mit wenig Arbeitsspeicher. Dabei werden die Modelle in mehreren Schritten in den Arbeitsspeicher geladen und anschließend in den GPU-Speicher übertragen. Dieses Verfahren reduziert die Menge des notwendigen Arbeitsspeichers, erhöht aber die Zeit, die zum Laden des Modells benötigt wird. Da im vorliegenden Fall genügend Arbeitsspeicher zur Verfügung stand, wurde auf dieses Verfahren verzichtet.

5.2.2 *Konfiguration der Trainingsdaten*

Für das Training des Modells Llama 2 wurde das Buch *Health Information Systems* von Winter u. a. (2023) im epub-Format in das Markdown-Format konvertiert. Die notwendigen Änderungen am Text sind in Abschnitt 4.2 beschrieben. Um diese Markdown-Datei in eine für das Modell verständliche Form umzuwandeln, wird die Bibliothek datasets (Lhoest u. a., 2021) von Huggingface verwendet. Sie ermöglicht

Parameter	Wert
max_train_samples	None
overwrite_cache	false
block_size	1024
validation_split_percentage	5
preprocessing_num_workers	1
keep_linebreaks	true

Tabelle 5.2: Parameter zur Auswahl und Konfiguration der Trainingsdaten

das Laden der Textdatei, die Umwandlung in Tokens und die Aufteilung in Blöcke. Die Parameter zur Auswahl und Konfiguration der Trainingsdaten sind in Tabelle 5.2 aufgelistet.

Der Parameter `train_file` ist nicht in der Tabelle dargestellt, beschreibt jedoch den Pfad zur genutzten Trainingsdatei. Diese Trainingsdatei wird einmal eingelesen und je nach Anzahl der unter Abschnitt 5.2.3 beschriebenen Epochen mehrfach verwendet.

`max_train_samples` beschreibt die maximale Anzahl von Blöcken, die aus der Trainingsdatei gelesen werden sollen. Zu Testzwecken kann hier eine geringere Anzahl an Blöcken verwendet werden, um die Konfiguration des Modells zu testen. Im finalen Training wurde dieser Parameter auf „None“ gesetzt, um alle Blöcke zu verwenden.

Der Parameter `overwrite_cache` beschreibt, ob der Text erneut in Tokens umgewandelt werden soll. Wenn sich der Text geändert hat, kann das Skript hier die nun tokenisierte Textdatei überschreiben.

Wie bereits erwähnt, wird der Text in Blöcke aufgeteilt. Jeder Block wird vom Modell vollständig und gleichzeitig gelesen. Die maximale Größe eines Blocks ist durch das Modell begrenzt und liegt bei Llama 2 bei 4096 Tokens. Standardmäßig verwenden Modelle jedoch eine Blockgröße von 1024 Tokens, weshalb hier diese Größe angenommen wird, wenn keine weiteren Angaben gemacht werden. Größere Blöcke führen zu einer schnelleren Verarbeitung des Textes und können zu einem besseren Verständnis der Zusammenhänge führen, da ein größerer Kontext betrachtet wird. Allerdings steigt mit der Größe der Blöcke auch die Fehleranfälligkeit, so dass teilweise auch kleinere Blöcke verwendet werden müssen, um ein Training erfolgreich durchzuführen. Probleme beim Training sind unter Abschnitt 5.2.7 beschrieben. Die Blockgröße kann mit dem Parameter `block_size` angepasst werden.

Der Parameter `validation_split_percentage` beschreibt den Anteil der Daten, der für die Validierung genutzt werden soll. Hier werden 5 % der Daten für die Validierung verwendet. Die Validierung liefert während des Trainings Informationen über die tatsächliche Leistung des Modells im Vergleich zum ungesehenen Text und dient dazu, den Fortschritt des Modells zu messen. Steigt der errechnete Fehlerwert des Validierungsdatensatz über einen größeren Zeitraum, ist davon auszugehen, dass das Modell beginnt den Status Overfitting zu erreichen. Ein Modell, welches einen sehr hohen Fehlerwert des Validierungsdatensatzes, aber einen sehr niedrigen Fehlerwert des Trainingsdatensatzes hat, kann keine ungesehenen Texte mehr verstehen und lediglich den Trainingsdatensatz zitieren. Die Analyse der Fehlerwerte über mehrere Epochen ist in Kapitel 6 aufgeführt.

`preprocessing_num_workers` beschreibt die Anzahl der Prozesse, die für die Umwandlung der Textdatei in Tokens verwendet werden sollen. Bei sehr großen Datenmengen ist die Umwandlung von Textdateien in Tokens eine sehr umfangreiche Aufgabe. Um diesen Vorgang zu beschleunigen, können mehrere Prozesse die Textdateien parallel übersetzen. In diesem Fall ist die Datenmenge jedoch klein genug, um die Aufgabe mit nur einem Prozess zu erledigen.

Der Parameter `keep_linebreaks` beschreibt, ob Zeilenumbrüche in der Textdatei erhalten bleiben sollen. In einigen Fällen können Textdateien viele Zeilenumbrüche enthalten, die der Formatierung des Textes dienen, aber dem Modell keine Informationen liefern oder es dazu veranlassen, diese Zeilenumbrüche zu imitieren. Aus diesem Grund können Zeilenumbrüche optional entfernt werden. Aufgrund der zuvor durchgeführten Datenkuration ist dies jedoch nicht notwendig, weshalb dieser Parameter auf „true“ gesetzt wird.

5.2.3 *Konfiguration des Trainings*

Die Konfiguration des Trainings orientiert sich an den „TrainingArguments“, die ein Teil der Transformers-Bibliothek sind. Nicht alle Parameter aus der Bibliothek müssen verwendet werden, weshalb hier nur abweichende Parameter vom Standard beschrieben werden. Die beschriebenen Parameter sind in Tabelle 5.3 aufgeführt.

Der Parameter `output_dir` definiert den Speicherpfad für die Ergebnisse des Trainings. Diese umfassen die trainierten Gewichte des Modells, eine abschließende Auswertung der Validierung, den Status des Trainers sowie gegebenenfalls während des Trainings erstellte Kontrollpunkte.

Parameter	Wert
output_dir	./trained/7B-3
overwrite_output_dir	true
do_train	true
do_eval	true
per_device_train_batch_size	1
per_device_eval_batch_size	1
evaluation_strategy	steps
eval_steps	50
learning_rate	3e−4
weight_decay	0,1
optim	adamw_torch
adam_beta1	0,9
adam_beta2	0,95
adam_epsilon	1e−5
max_grad_norm	1,0
num_train_epochs	3
lr_scheduler_type	cosine
warmup_steps	0
save_strategy	steps
save_steps	100
save_total_limit	1
no_cuda	false
seed	42
fp16	true | false
bf16	false | true
half_precision_backend	auto
ddp_backend	nccl
deepspeed	./ds_configs/stage2_offload.json

Tabelle 5.3: Parameter zur Konfiguration des Trainings. Die Parameter `fp16` und `bf16` können entweder `true` und `false` sein, wenn ein Training im „FP16“-Datenformat bei der Benutzung von Nvidia Tesla V100 Grafikkarten gewählt wird, oder als `false` und `true` gewählt werden für ein Training im „BF16“ Format bei der Benutzung von Nvidia Tesla A30 Grafikkarten.

Der Parameter `overwrite_output_dir` legt fest, ob der Ergebnis-Ordner überschrieben werden soll, wenn er bereits existiert. Diese Option bestimmt auch, ob das Training von einem zuvor erstellten Kontrollpunkt aus fortgesetzt werden soll. Wenn der Ergebnis-Ordner überschrieben wird, kann von keinem Kontrollpunkt aus fortgefahren werden.

Die Parameter `do_train` und `do_eval` geben an, ob das Training und die Validierung durchgeführt werden sollen. Falls keine Validierung durchgeführt wird, wird der Trainingsdatensatz dennoch in Trainings- und Validierungsdatensatz aufgeteilt.

Während des Trainings und der Validierung werden die Fehlerfunktionen von mehreren Blöcken berechnet und danach gemittelt. Daraufhin wird ein Gradient basierend auf dieser kumulativen Fehlerfunktion berechnet, welcher die Gewichte des Modells anpasst. Eine detailliertere Beschreibung des Ablaufs findet sich in Abschnitt 2.2.3.1. Die Anzahl der Blöcke pro „Batch“ (siehe Definition 28) wird durch die Parameter `per_device_train_batch_size` und `per_device_eval_batch_size` festgelegt. Wenn man eine Batch-Größe von 1 und 3 GPUs verwendet, erhält man somit 3 Blöcke pro Gradientenberechnung. Während höhere Batch-Größen die Berechnungszeit reduzieren können, führen sie auch zu ungenaueren Gradienten und erfordern mehr Speicher auf der GPU. Die Batch-Größe ist in diesem Fall auf 1 gesetzt, da die verwendeten Nvidia V100-GPUs als auch Nvidia A30-GPUs nur genug Speicher für eine Batch-Größe von 1 hatten.

Die Parameter `evaluation_strategy` und `eval_steps` legen fest, in welchen Intervallen die Validierung durchgeführt werden soll. In diesem Szenario wird die Validierung alle 50 Iterationen durchgeführt. Eine Iteration bezieht sich auf die Berechnung eines Gradienten.

Die Lernrate des Modells wird durch den Parameter `learning_rate` beschrieben. Dieser Parameter bestimmt, in welchem Ausmaß die Gewichte des Modells angepasst werden. Eine sehr hohe Lernrate kann dazu führen, dass das Modell nicht konvergiert, während eine sehr niedrige Lernrate zu langen Trainingszeiten führt. Die Lernrate wurde aus dem Artikel über Llama-Modelle von Touvron u. a. (2023a) übernommen.

In dem Artikel zu den Llama-Modellen von Touvron u. a. (2023a) wird ebenfalls ein Gewichtsabnahme-Wert von $0,1$ verwendet, der auch bei diesem Training mithilfe des Parameters `weight_decay` eingestellt wurde. Die Gewichtsabnahme führt zu einer kontinuierlichen Verringerung der Gewichte des Modells. Dadurch soll verhindert werden, dass es sich zu stark an einzelne Trainingsdaten anpasst.

Die Parameter `adam_beta1`, `adam_beta2` und `adam_epsilon` beschreiben die Parameter des verwendeten AdamW-Optimierers, der im Parameter `optim` eingestellt ist. Diese Werte entsprechen ebenfalls den Werten, die im Artikel zu den Llama-Modellen von Touvron u. a. (2023a) angegeben sind. AdamW ist in verschiedenen Implementierungen verfügbar. Hier wurde die neueste Implementierung der PyTorch-Bibliothek verwendet. Die Funktionsweise des AdamW-Optimierers ist im Artikel von Loshchilov und Hutter (2019) genauer beschrieben.

Der Parameter `max_grad_norm` beschreibt die maximale Norm des Gradienten (siehe Abschnitt 2.2.3.1), die durch die Gewichte des Modells nicht überschritten werden darf. Eine weitere Bezeichnung, die in der Bibliothek DeepSpeed oder in Artikeln zu Llama-Modellen genutzt wird, ist `gradient clipping`. Er begrenzt die Größe des Gradienten, der in großen neuronalen Netzen explosionsartig ansteigen kann. Zu große Gradienten führen zu schlechteren Trainingsergebnissen und zu einer zu starken Anpassung der Gewichte, was wiederum zu einer Oszillation um ein Minimum führt. Um die Gradienten zu begrenzen, wird die L2-Norm des Gradienten berechnet[2]. Wenn diese Norm den angegebenen Maximalwert überschreitet, wird der Gradient herunter skaliert, bis er die Maximalnorm nicht mehr überschreitet.

Der Parameter `num_train_epochs` gibt die Anzahl der zu trainierenden Epochen an. Eine Epoche umfasst einen vollständigen Durchlauf des Trainingsdatensatzes. Mehr Epochen können insbesondere bei kleineren Datensätzen zu besseren Ergebnissen führen, da sich das Modell besser an die Trainingsdaten anpassen kann. Zu viele Epochen führen zu einer Überanpassung. In dieser Arbeit wurde das Modell Llama 2 7B mit einer, drei, fünf und zehn Epochen trainiert. Bei der Verwendung von Nvidia V100 Grafikkarten traten während des Trainings Probleme auf, die in Abschnitt 5.2.7 genauer beschrieben sind.

Durch die Anwendung der Parameter `lr_scheduler` und `warmup_steps` kann eine Aufwärmphase des Trainings realisiert werden. Die Aufwärmphase bezeichnet den Start des Trainings, in dem die Lernrate der `lr_scheduler`-Funktion innerhalb der ersten `warmup_steps` Iterationen schrittweise von 0 auf den gewünschten Wert erhöht wird. Dieser Ansatz verhindert eine zu schnelle Anpassung der Modellgewichte an spezielle Details der Trainingsdaten. Vor allem bei untrainierten Modellen werden während des Trainings diese erlernten Fehler wieder korrigiert. Dies führt jedoch ohne eine Aufwärmphase zu längeren Trainingszeiten und schlechteren Ergebnissen. In diesem

2 Die L2-Norm, auch bezeichnet als euklidische Norm, entspricht der pythagoreischen Länge eines Vektors vom Ursprung. Gegeben eines Vektors (a, b, c) entspräche die L2-Norm $\sqrt{a^2 + b^2 + c^2}$

Fall kann die Aufwärmphase übersprungen werden, da ein bereits vortrainiertes Modell genutzt wird.

Die Parameter `save_strategy`, `save_steps` und `save_total_limit` beschreiben, wie oft und in welchem Abstand Modelle während des Trainings gespeichert werden sollen. Hier wird alle 100 Iterationen das Modell gespeichert, wobei maximal 1 Kontrollpunkt gleichzeitig existiert. Kontrollpunkte ermöglichen es, abgebrochene oder fehlgeschlagene Trainingsläufe wieder aufzunehmen. Bei längerem Training kann die maximale Laufzeit der Skripte erreicht werden. Eine Wiederaufnahme des Trainings an diesem Punkt ist dann möglich. Das Rechenzentrum beschränkt die maximale Laufzeit von Skripten auf 2 Tage. Wenn das Training länger als 2 Tage dauert, kann diese Grenze überschritten werden. Die Begrenzung ist notwendig aufgrund der gemeinsamen Nutzung des Rechenzentrums. Durch die regelmäßige Unterbrechung von Skripten können andere Skripte in der Warteschlange zwischengeschoben werden. Zusammen mit der Begrenzung ergibt sich im Allgemeinen eine Wartezeit von maximal 2 Tagen. Durch die Einführung von Kontrollpunkten im Training führt eine Unterbrechung nicht zu einem Rückschlag im Fortschritt.

Der Begriff `no_cuda` beschreibt die Durchführung des Trainings ohne die Verwendung einer GPU. Diese Einstellung wird für Testzwecke verwendet, damit Testskripte auf Systemen ausgeführt werden können, die nicht über die erforderliche Anzahl von GPUs verfügen.

Die Verwendung des Parameters `seed` beeinflusst die Zufälligkeit des Trainingsprozesses. Während des Trainingsprozesses werden einige Zufallsvariablen verwendet, um Werte zu initialisieren. Jedoch ist die Auswirkung im Kontext des Fortführenden Trainings irrelevant. Es wird jedoch ein fester Wert gesetzt, da das Trainingsskript auch für untrainierte Modelle genutzt werden kann. Mit Hilfe eines Seeds erstellen Zufallszahlengeneratoren zufällig verteilte, jedoch reproduzierbare Zahlenfolgen. Daher sind Modelle mit gleichem Seed und Datensatz identisch.

Zur Konfiguration der verwendeten Datentypen stehen neben dem oben genannten `torch_dtype` die beiden Parameter `fp16` und `bf16` zur Verfügung. Während `torch_dtype` die Initialisierung des Modells beschreibt, beeinflussen diese Parameter die Datentypen während des Trainings. Es kann nur eine der beiden Optionen verwendet werden. Im Falle des Trainings mit Nvidia V100 Grafikkarten und der ZeRO Stufe 2 Optimierung wurde FP16 aufgrund von Architekturanforderungen verwendet. Beim Training mit Nvidia A30 Grafikkarten und ZeRO Stufe 3 Optimierung wurde BF16 verwendet. Die Zahl 16 steht hier für die Anzahl der Bits pro Wert, was wiederum einen großen

Einfluss auf den Speicherbedarf und die Genauigkeit des Modells hat. Datentypen sollten nicht geändert werden, wenn ein vortrainiertes Modell weiter trainiert wird, da eine Änderung schnell zu Fehlverhalten führen kann. Die Llama 2 Modelle wurden mit dem Datentyp „bf16“ (ausgeschrieben BrainFloat 16) trainiert. Dieser Datentyp ist nur auf bestimmten GPU-Architekturen verfügbar. Ein Datum im BFloat16-Format hat eine kleinere Mantisse (7 Bits) als ein Float16-Datum (10 Bits Mantisse). Dies führt zu einer geringeren Genauigkeit, verkürzt jedoch die Zeit, die für die Konvergenz des Modells benötigt wird. Die verwendeten V100 Grafikkarten haben nicht die notwendige Architektur, um BFloat16 Werte zu unterstützen, daher wurde das Llama 2 Modell hier in ein Float16 Format konvertiert. Dies führt zu Problemen beim Training und beschränkt die maximale Anzahl der Epochen auf 3. Eine genauere Erklärung der Probleme ist in Abschnitt 5.2.7 beschrieben. Bei der Verwendung von A30-Grafikkarten ist die Nutzung von BFloat16-Werten möglich. Mit diesem Datentyp wurde das Modell zusätzlich auf 5 und 10 Epochen trainiert.

Die Parameter `half_precision_backend` und `ddp_backend` beschreiben Architekturen für die Ausführung des Trainings mit Hilfe der Bibliothek Transformers. Während `half_precision_backend` die Architektur für die Ausführung von Trainingsschritten mit Float16-Werten festlegt, beschreibt `ddp_backend` die Kommunikationsarchitektur für den Datenaustausch zwischen GPUs. Die DeepSpeed Bibliothek verwendet die „NCCL“ Architektur, daher wird diese hier eingestellt.

Die DeepSpeed Konfiguration wird separat in einer JSON Datei gespeichert. Durch Setzen des Parameters `deepspeed` verwendet die Transformers Bibliothek (Wolf u. a., 2020) die DeepSpeed Bibliothek (Rajbhandari u. a., 2020) zur Ausführung des Trainings. Wichtig bei der Verwendung von DeepSpeed mit Transformers ist eine identische Konfiguration der Trainingsparameter, wie zum Beispiel `batch_size`, `gradient_accumulation_steps` und `learning_rate`.

5.2.4 *Konfiguration des DeepSpeed Trainings*

Die DeepSpeed-Konfiguration ist in Tabelle 5.4 beschrieben. Die hier verwendeten Parameter beschreiben ein Training mit ZeRO Stufe 2 Optimierung und zusätzlichem CPU Offloading. Die ZeRO Stufe 2 Optimierung beschreibt einen Algorithmus zur Reduzierung des Speicherbedarfs während des Trainings. Dieser Algorithmus wird in Rajbhandari u. a. (2020) genauer beschrieben. Zusätzlich zu dieser Optimierung werden Teile der Berechnungen auf die CPU ausgelagert, um den Speicherbedarf pro GPU weiter zu reduzieren. Diese Konfiguration wurde beim Training der Modelle mit Nvidia Tesla V100 Grafikkar-

Parameter	Wert
fp16	
enabled	auto
loss_scale	0
loss_scale_window	1000
initial_scale_power	16
hysteresis	2
min_loss_scale	1
bf16	
enabled	true | false
optimizer	
type	AdamW
lr	auto
betas	auto
eps	auto
weight_decay	auto
scheduler	
type	WarmupLR
warmup_min_lr	auto
warmup_max_lr	auto
warmup_num_steps	auto
zero_optimizations	
stage	2
contiguous_gradients	true
overlap_comm	true
reduce_scatter	true
reduce_bucket_size	2e8
allgather_bucket_size	2e8
offload_optimizer	
device	cpu
pin_memory	true
gradient_clipping	1
steps_per_print	500
wall_clock_breakdown	false
train_micro_batch_size_per_gpu	auto

Tabelle 5.4: DeepSpeed Konfiguration mit Benutzung der ZeRO Stufe 2 Optimierung. Für diese Stufe wurde der Parameter `enabled` unter der Rubrik `bf16` auf „false“ gesetzt. Für die Nutzung der ZeRO Stufe 3 Optimierung ist dieser Wert auf „true“ gesetzt.

Parameter	Wert
zero_optimization	
stage	3
contiguous_gradients	true
stage3_max_live_parameter	1e9
stage3_max_resuse_distance	1e9
stage3_prefetch_bucket_size	1e7
stage3_param_persistence_threshold	1e5
reduce_bucket_size	1e7
sub_group_size	1e9
offload_param	
device	cpu
pin_memory	true

Tabelle 5.5: DeepSpeed ZeRO Stufe 3 Konfiguration

ten verwendet. Tabelle 5.5 beschreibt zusätzlich eine Konfiguration des `zero_optimization` Abschnitts bei Verwendung der ZeRO Stufe 3 Optimierung. Hier werden zusätzlich Parameter der Modelle auf die CPU ausgelagert. Diese Konfiguration wurde beim Training der Modelle mit Nvidia Tesla A30 Grafikkarten verwendet.

Der Abschnitt `fp16` beschreibt den Umgang mit Float16-Werten während des Trainings. Dabei handelt es sich um eine dynamische Skalierung der Fehlerwerte (engl. „Dynamic Loss Scaling“). Die Skalierung der Fehlerwerte ist notwendig, da aufgrund der geringeren Genauigkeit von Float16-Werten kleinere Werte der Fehlerwerte gerundet werden und verloren gehen. Aus diesem Grund werden die Fehlerwerte bei der Berechnung um mehrere Potenzen skaliert. Diese Skalierung kann auch zu einem Überlauf über den Wertebereich des Float16 Datentyps führen. DeepSpeed verwendet hier eine automatische, dynamische Skalierung der Fehlerwerte, ohne einen Überlauf zu verursachen. Eine genauere Erklärung der Skalierung von Fehlerwerten findet sich im Artikel von Micikevicius u. a. (2018).

Der Parameter `loss_scale` beschreibt die konstante Skalierung der Fehlerwerte. Ist er auf 0 gesetzt, wird eine dynamische Skalierung verwendet. Der Parameter `loss_scale_window` beschreibt das Werteintervall, in dem die dynamische Skalierung erfolgt. Der Parameter `initial_scale_power` beschreibt die Größe der initialen Skalierung der Fehlerwerte. Die tatsächliche Skalierung der Fehlerwerte entspricht $2^{initial\_scale\_power}$. Der Parameter `hysteresis` beschreibt die

minimale Anzahl von Schritten, in denen die Skalierung nicht verändert werden kann. Der Parameter `min_loss_scale` beschreibt die minimale Skalierung der Fehlerwerte. Hier entspricht der Wert 1 keiner Skalierung.

Der Abschnitt `bf16` beschreibt die Verwendung des binären Datentyps BF16 (ausgeschrieben BrainFloat 16) während des Trainings. Ist er auf `true` gesetzt, so werden die Berechnungen mit dem Datentyp BFloat16 durchgeführt. Ist er auf `false` gesetzt, so werden die Berechnungen mit dem Datentyp Float16 durchgeführt. Bei einem Training mit V100 Grafikkarten und ZeRO Stufe 2 Optimierung ist dieser Parameter auf „false“ gesetzt. Im Falle des Trainings mit A30 Grafikkarten und einer ZeRO Stufe 3 Optimierung ist dieser Parameter auf „true“ gesetzt.

Der Abschnitt `optimizer` beschreibt die Parameter des Optimierungsalgorithmus. Wie bereits bei den Trainingsparametern beschrieben, wird der AdamW-Optimierer verwendet. Die Parameter `lr`, `betas`, `eps` und `weight_decay` müssen mit den in den Trainingsparametern eingestellten Werten übereinstimmen. Aus diesem Grund werden diese Parameter vor Beginn des Trainings auf „auto“ gesetzt und durch die Transformers-Bibliothek ergänzt.

Der Abschnitt `scheduler` beschreibt die Parameter des Lernratenplaners. Der Lernratenplaner ist für die Anpassung der Lernrate während des Trainings verantwortlich. Er dient in diesem Fall zur Durchführung der Aufwärmphase. Ist die Aufwärmphase in den Trainingsargumenten deaktiviert, so wird die Bibliothek diese Phase auch in der DeepSpeed-Konfiguration deaktivieren.

Der Abschnitt `zero_optimizations` beschreibt die Parameter der ZeRO Stufe 2 Optimierung (Rajbhandari u. a., 2020). Der Parameter `contiguous_gradients` beschreibt die Auslagerung der Gradienten in einen zusammenhängenden Speicherbereich während der Berechnung. Dadurch wird eine Fragmentierung des Speichers während der Backpropagation vermieden. Mit Hilfe des Parameters `overlap_comm` wird versucht, die Gradienten während der Berechnung der Backpropagation zu reduzieren, um eine schnellere Gesamtberechnung zu ermöglichen. `reduce_scatter` beschreibt die Verwendung einer speziellen Reduktionsmethode „Reduce Scatter“ zur Mittelung von Gradienten. In Kombination mit den Parametern `reduce_bucket_size` und `allgather_bucket_size` wird die maximale Anzahl der in einem Schritt zu reduzierenden Gradienten festgelegt. Diese Option reduziert den Speicherbedarf während des Trainings erheblich. Mit Hilfe der Option `offload_optimizer` wird der Zustand und die Berechnung des Optimierers auf die CPU ausgelagert. Optional kann dies für sehr

große Modelle auch auf eine NVMe SSD erfolgen. Mit Hilfe der Einstellung `pin_memory` wird der für die Berechnung benötigte Speicher auf der CPU reserviert. Dies führt zu einer besseren Performance, aber auch zu zusätzlichem Speicherbedarf.

Wie bereits bei den Trainingsargumenten durch den Parameter `max_grad_norm` beschrieben, wird die maximale Größe des Gradienten durch den Parameter `gradient_clipping` festgelegt. Eine automatische Übernahme der Konfiguration der Trainingsargumente ist in diesem Fall nicht möglich. Die Werte müssen dennoch übereinstimmen.

Die Parameter `steps_per_print` und `wall_clock_breakdown` beschreiben die Ausgabe von Informationen während des Trainings. Zusätzlich zur Ausgabe der Transformers-Bibliothek werden alle 500 Iterationen weitere Informationen der DeepSpeed-Bibliothek ausgegeben. Mit Hilfe des Parameters `wall_clock_breakdown` wird zusätzlich die Messung der verstrichenen Zeit für jede Phase einer Iteration ausgegeben.

Der Parameter `train_micro_batch_size_per_gpu` beschreibt die Anzahl der Batches pro GPU. Dieser Wert wird aus den Trainingsargumenten übernommen.

Für die Ausführung des Trainings mit der ZeRO Stufe 3 Optimierung auf Nvidia Tesla A30 Grafikkarten wurde der Abschnitt `zero_optimization` in Tabelle 5.4 durch die in Tabelle 5.5 gezeigten Parameter ersetzt. Weggefallene Parameter sind:

- `overlap_comm`
- `reduce_scatter`
- `allgather_bucket_size`

Zusätzlich hinzugekommene Parameter sind:

- `stage3_max_live_params`
- `stage3_max_reuse_distance`
- `stage3_prefetch_bucket_size`
- `stage3_param_persistence_threshold`
- `sub_group_size`

sowie der Abschnitt `offload_param`. Der Parameter `stage3_max_live_params` beschreibt die maximale Anzahl von Gewichten, die während des Trainings in den Speicher der GPU geladen werden. Kleinere Werte

Bibliothek	Version
datasets	2.13.1
DeepSpeed	0.10.0
evaluate	0.4.0
PyTorch	2.0.1
sentencepiece	0.1.96
Transformers	4.31.0

Tabelle 5.6: Verwendete Bibliotheken zum Training

führen zu weniger Speicherverbrauch, aber auch zu mehr Kommunikation. Der Parameter `stage3_max_reuse_distance` beschreibt, dass ein Gewicht nicht freigegeben werden kann, wenn es innerhalb dieser Anzahl von Gewichten erneut zur Berechnung verwendet wird. Auch hier gilt, dass kleinere Werte zu einer geringeren Speicherbelegung, aber auch zu einer erhöhten Kommunikation führen. Der Parameter `stage3_prefetch_bucket_size` beschreibt die Anzahl der Gewichte, die während des Trainings im Voraus geladen werden. Der Parameter `stage3_param_persistence_threshold` verhindert die Partitionierung von Gewichten, die kleiner als dieser Wert sind. Der Parameter `sub_group_size` beschreibt die maximale Anzahl von Parametern, die gleichzeitig für die Berechnung verwendet werden.

5.2.5 *Verwendete Bibliotheken zum Training*

Die für das Training verwendeten Bibliotheken sind in Tabelle 5.6 aufgelistet. Einige der hier aufgeführten Pakete benötigen weitere Abhängigkeiten, die über die verwendete Paketverwaltung Package Installer for Python (PIP)[3] installiert werden.

Zur Verwaltung der Datensätze wird die Bibliothek „datasets“ verwendet (Lhoest u. a., 2021). Mit Hilfe dieser Bibliothek können Textdateien während des Trainings geladen, in Tokens umgewandelt und in Blöcke gruppiert werden. Über eine Schnittstelle zu den Bibliotheken Transformers und DeepSpeed ermöglicht „datasets“ auch die geordnete Zuordnung von Blöcken zu verschiedenen GPUs.

Die Bibliothek „DeepSpeed“ wird zur Beschleunigung des Trainings und der Ausführung auf mehreren Grafikkarten verwendet. Sie ermöglicht die Nutzung der ZeRO-Optimierung, die es erlaubt, Teile des Trainings und des Modells auf mehrere GPUs, der CPU und lokalen NVMe SSDs auszulagern. Der Prozess dieser Optimierung und die

3 `https://github.com/pypa/pip` abgerufen am 19.8.2023

Cluster	Knoten	Grafikkarten	GPU-RAM
Paula	12	8 Nvidia Tesla A30	24 GB
Clara	8	4 Nvidia Tesla V100	32 GB
Clara	23	8 Nvidia GeForce RTX 2080 Ti	11 GB

Tabelle 5.7: GPU Werte des des Rechenzentrums der Universität Leipzig

Einführung dieser Bibliothek werden in Rajbhandari u. a. (2020) genauer beschrieben.

Die Bibliothek „evaluate“ wird zur Berechnung der Modellgenauigkeit verwendet. Sie stammt von Huggingface[4].

Die Bibliothek „PyTorch“ bildet die Grundlage für die Berechnung der neuronalen Netze. Sie ermöglicht neben der eigentlichen Berechnung aller Trainingsphasen auch die Verwendung von GPUs (Paszke u. a., 2019).

Die Bibliothek „sentencepiece“ wird für die Tokenisierung der Texte genutzt. Sie wird im Skript nicht explizit verwendet, ist hier aber als zusätzliche Voraussetzung für die Verwendung des Tokenizers des Llama-Modells aufgeführt. Diese Bibliothek wurde auch im Artikel von Touvron u. a. (2023a) verwendet und wurde von Google veröffentlicht (Kudo und Richardson, 2018).

Die Bibliothek „Transformers“ wird für die Verwaltung der Modelle und die Berechnung der neuronalen Netze verwendet. Sie ermöglicht die Verwendung vortrainierter Modelle und bietet einen abstrakten Trainer, der die verschiedenen Phasen des Trainings steuert. Außerdem erlaubt sie die Verwendung der Bibliothek „DeepSpeed“ ohne zusätzliche Anpassung. Die Bibliothek wurde in Wolf u. a. (2020) vorgestellt.

5.2.6 *Training auf einem GPU Computing Cluster*

Das Trainieren der Modelle ist wesentlich rechenintensiver als deren Anwendung. Aus diesem Grund kann ein normaler Computer dafür nicht verwendet werden. Stattdessen wurde ein GPU Computing Cluster des Rechenzentrums der Universität Leipzig verwendet.

Das Rechenzentrum der Universität Leipzig stellt zwei GPU Computing Cluster mit den Namen „Paula“ und „Clara“ zur Verfügung.

4 `https://github.com/huggingface/evaluate` abgerufen am 19.8.2023

Cluster	CPU	RAM
Paula	2 AMD(R) EPYC(R) 7713	1 TB
Clara	1 AMD(R) EPYC(R) 7551P	512 GB

Tabelle 5.8: CPU Werte des Rechenzentrums der Universität Leipzig

Die technischen Daten der beiden Cluster sind in Tabelle 5.7 und Tabelle 5.8 aufgelistet. Das „Clara"-Cluster ist hierbei zweigeteilt, da einige Knoten mit Nvidia Tesla V100 Grafikkarten und andere mit Nvidia GeForce RTX 2080 Ti Grafikkarten ausgestattet sind. Alle Knoten innerhalb eines Clusters sind über ein 100 Gbit/s Infiniband-Netzwerk miteinander verbunden. Die Verbindung zum Internet erfolgt über eine 25 Gbit/s Ethernet-Verbindung.

Die Ausführung der Skripte erfolgt mit Hilfe der Software „Slurm Workload Manager"[5]. Diese ermöglicht eine umfangreiche Reservierung von Ressourcen und bietet eine Bash-Schnittstelle zur Konfiguration und Ausführung der Skripte. Mit Hilfe von Slurm-Bash-Dateien konnte das Training der Modelle auf dem Cluster durchgeführt werden.

Während des Trainings wurde durchschnittlich alle 12 Sekunden eine Iteration des Modells berechnet. Eine Iteration entspricht einer vollständigen Berechnung der Ausgaben über einen Batch und einer vollständigen Anpassung aller Gewichte (engl. „Forwards-" und „Backwardspass"). Eingeschlossen sind auch alle Kommunikationen, die zwischen den GPUs durchgeführt werden, um ihren aktuellen Zustand untereinander abzustimmen. Diese Geschwindigkeit ist unabhängig von der Blockgröße. Allerdings hängt die Länge des Trainings von der Anzahl der Iterationen ab, daher von der Anzahl der Blöcke und somit von der Blockgröße. Der kurierte Datensatz, der das Buch Winter u. a. (2023) repräsentiert, umfasst ca. 34.500 Tokens. Dementsprechend wird eine Epoche bei einer Standardblockgröße von 1024 Tokens in 34 Schritten beziehungsweise 6:14 Minuten berechnet. Eine größere Blockgröße war aufgrund von Speicherfehlern nicht möglich.

Bei der Ausführung des Trainings auf Nvidia v100 GPUs traten in unregelmäßigen Abständen Fehler auf, die in Abschnitt 5.2.7 genauer beschrieben sind. Diese konnten durch eine Reduzierung der Blockgröße auf 256 Tokens minimiert werden. Eine Epoche bestand somit aus 134 Schritten und dauerte 24:34 Minuten.

5 `https://slurm.schedmd.com/documentation.html` abgerufen am 24.8.2023

Epoche	Grafikkarten	Bezeichnung
1	4 Nvidia Tesla V100	llama_1e_v100
1	4 Nvidia Tesla A30	llama_1e_a30
3	4 Nvidia Tesla V100	llama_3e_v100
3	4 Nvidia Tesla A30	llama_3e_a30
5	4 Nvidia Tesla A30	llama_5e_a30
10	4 Nvidia Tesla A30	llama_10e_a30

Tabelle 5.9: Trainierte Llama 2 7B Modelle. Das Training wurde von dem vortrainierten Llama 2 7B Huggingface Modell begonnen.

Insgesamt wurden 6 Modelle trainiert und 8 Modelle evaluiert. Tabelle 5.9 zeigt die in dieser Arbeit trainierten Modelle.

Zusätzlich evaluiert wurden die Modelle:

- GPT4 Modell: „gpt4“
- Llama 2 7B untrainiert: „llama_0e“

5.2.7 *Probleme während des Trainings*

Die Durchführung des Trainings verlief nicht ohne Probleme. Ideal wäre es, mehrere Modelle mit unterschiedlicher Epochenzahl und Größe zu trainieren. Neben dem Modell Llama 2 7B verspricht ein größeres Modell wie die Modelle 13B oder 70B deutlich bessere Leistungen sowohl in der Textnachahmung als auch im Wissensverständnis und damit in den Ergebnissen der Evaluation. Das Training wurde auf 4 GPUs pro Knoten mit jeweils 32 GB GPU-Random Access Memory (RAM) bzw. 24 GB durchgeführt, wie bereits unter Abschnitt 5.2.6 erwähnt. Um größere Modelle zu trainieren, muss diese Anzahl erhöht werden. Die Trainingsparameter erlauben hier eine einfache Erweiterung auf zusätzliche Grafikkarten. Die Trainingsskripte müssen dafür nicht angepasst werden.

Das Training auf mehreren Knoten mit mehr Grafikkarten wurde testweise durchgeführt. Dazu wurde ein Testskript von Huggingface, das die Kommunikation zwischen Grafikkarten und Knoten sicherstellt, und das Trainingsskript verwendet. Beide Skripte sind in dieser Konfiguration fehlgeschlagen. Der Grund dafür ist ein Kommunikationsfehler zwischen den Knoten. Die Kommunikation während des Trainings wird vollständig von den Bibliotheken Transformers und DeepSpeed übernommen. Dazu wird ein verwendeter Knoten als Main-Knoten definiert, der das Nachrichten- und Daten-Routing übernimmt. Andere Knoten übernehmen die Rolle von Client-Knoten. Auf

dem Main-Knoten wird ein Server gestartet, mit dem sich die Client-Knoten über IPv4 oder IPv6 verbinden können. Diese Verbindung schlug in allen Einstellungen fehl. Eine Lösung für dieses Problem konnte nicht gefunden werden. Aus diesem Grund wurde das Training auf einem Knoten und damit 4 GPUs beschränkt.

Größere Modelle wie das 70B Modell können mit dem aktuellen Setup nicht trainiert werden. Aus diesem Grund und um Ressourcen zu sparen, wurde das Training mit dem Modell Llama 2 7B begonnen. Bei der Ausführung auf Nvidia V100 Grafikkarten traten in unregelmäßigen Abständen Überlauffehler auf. Wie in Abschnitt 5.2.4 beschrieben, werden die berechneten Fehlerwerte der Fehlerrückkopplungsfunktion während der Berechnung skaliert, um eine höhere Genauigkeit zu erreichen und einen Unterlauf zu vermeiden. Diese Skalierung führte zu einem Überlauf über den Maximalwert des Datentyps Float16. Da in der DeepSpeed-Konfiguration eine dynamische Fehlerskalierung eingestellt war, wurde die Skalierung anschließend um eine Potenz reduziert. Dieser Vorgang wiederholte sich in unregelmäßigen Abständen und führte schließlich zum Erreichen des minimalen Skalierungsfaktors 1, der keiner Skalierung entspricht. Ist dieser Wert erreicht und eine Skalierung dennoch erforderlich, wird das Training abgebrochen.

Die Gründe für die aufgetretenen Skalierungsfehler sind nicht bekannt. Eine mögliche Ursache ist die Verwendung des Llama-Modells mit dem Datentyp Float16. Dieses Modell wurde ursprünglich mit dem Datentyp BFloat16 trainiert. Eine Transformation in Float16-Werte wird auch von Huggingface nicht empfohlen. Ebenso wird ein Fehler in der Implementierung von DeepSpeed vermutet, der das Laden von Modellen im Float16-Format nicht korrekt umsetzt. Im offiziellen GitHub Repository von DeepSpeed wurde bereits eine Lösung vorgeschlagen, die jedoch noch nicht in DeepSpeed übernommen wurde. Die Verwendung von BFloat16 Werten ist nur auf dem Cluster „Paula“ mit Nvidia Tesla A30 Grafikkarten möglich. BFloat16 Werte können nur von Grafikkarten umgesetzt werden, die die AMP (Automatic Mixed Precision) Architektur unterstützen. Diese Architektur wird von den Grafikkarten der Nvidia A100 Serie unterstützt, die für das initiale Training der Llama Modelle verwendet wurden. Im Rechenzentrum stehen jedoch nur Grafikkarten der Serien Nvidia Tesla V100, Nvidia GeForce RTX 2080 Ti und Nvidia Tesla A30 zur Verfügung. Sowohl die V100- als auch die RTX2080-Serie unterstützen die Verwendung von BFloat16-Werten nicht. Die A30-Serie unterstützt zwar BFloat16, bietet aber nur 24 GB RAM, weshalb hier eine ZeRO Stufe 3 Optimierung gewählt wurde. Bei der Verwendung von A30-Grafikkarten traten bei einem Training von bis zu 10 Epochen keine Skalierungsfehler auf.

Anfangs verhinderte der Skalierungsfehler das Training der Modelle auf V100-Grafikkarten vollständig. Durch Anpassung der Blockgröße in Abschnitt 5.2.3 auf kleinere Werte konnte die Häufigkeit der Skalierungsfehler minimiert werden, so dass mit dieser Einstellung ein Training von bis zu 3 Epochen möglich war.

Während des Trainings traten immer wieder Speicherfehler auf, die anzeigten, dass der gesamte Speicher der Grafikkarte belegt war, jedoch mehr benötigt wurde. Diese Fehler waren unabhängig vom Grafikkartentyp und der Konfiguration des Trainings. Erst die Minimierung der Batchgröße auf 1 pro GPU, die Aktivierung des CPU-Offloads und die Nutzung von 256 GB CPU RAM ermöglichte ein erfolgreiches Training des Modells Llama 2 7B. Dieser Speicherbedarf ist unerwartet hoch und kann nicht durch die Größe des Modells erklärt werden.

5.3 GENERIERUNG VON ANTWORTEN

Die Generierung von Antworten auf die erstellten Evaluationsdaten erfolgte auf drei verschiedene Arten. Für die Textgenerierung durch das GPT4-Modell wurde die offizielle Webseite der OpenAI Foundation verwendet[6]. Die Verwendung des untrainierten Llama-Modells sowie der trainierten Modelle mit V100-Grafikkarten erfolgte mit Hilfe eines eigens erstellten Skripts. Dieses Skript wurde größtenteils für die Nutzung der mit A30-Grafikkarten trainierten Modelle verwendet, musste jedoch modifiziert werden, da die Generierung ausschließlich auf einer GPU erfolgt, die A30-Grafikkarten jedoch mit 24 GB RAM das Modell nicht vollständig unterstützen konnten. Aus diesem Grund wurde das Modell im 8-Bit-Modus geladen. Dieser spezielle Modus reduziert alle Gewichte des Modells auf 8-Bit-Floatwerte, was zu einer geringeren Genauigkeit, aber auch zu einem geringeren Speicherbedarf führt. Eigentlich sollte das 7B Modell mit 14 GB Speicherbedarf ohne Datentyptransformierung mit einer A30-Grafikkarte genutzt werden können, jedoch wurde in diesem Aufbau das Speicherlimit der Grafikkarte erreicht. Ein Grund dafür ist nicht bekannt.

5.3.1 *Erstellung des Evaluierungsdatensatzes*

Vor der Generierung der Antworten mussten die in den jeweiligen Klausuren gestellten Fragen aus Abschnitt 4.4 transformiert werden. Dazu wurden diese Fragen um notwendigen Kontext ergänzt, wenn dieser in der Frage angesprochen wurde oder auf den Ergebnissen vorhergehender Fragen aufbaute. Zusätzlich wurden alle Fragen in die englische Sprache übersetzt, da das Llama-Modell fast ausschließlich auf englischen Texten trainiert wurde und das verwendete Buch

6 `https://chat.openai.com/` abgerufen am 19.8.2023

```
Instruction: You are given a question and a context. Answer ↙
    the question to your best knowledge.
Question: <transformed>
Context: <context>
Answer:
```

Abbildung 5.1: Struktur einer Eingabe für das Llama-Modell

Winter u. a. (2023) ebenfalls in englischer Sprache verfasst ist. Jeder Frage wurden 6 Eigenschaften zugeordnet.

- **question**: Die ursprüngliche Frage aus der Quelle
- **transformed**: Die umformulierte und eventuell übersetzte Frage
- **true_answer**: Die erwartete richtige Antwort
- **num_answers**: Die Anzahl der in der erwarteten richtigen Antwort enthaltenen Antworten
- **source**: Abkürzung für die Quelle der Frage
- **context**: Ein optionaler Kontext, der zur Beantwortung der Frage benötigt wird

Mit Hilfe dieser Struktur wurden die Fragen in eine weitere explizite Frageform umgewandelt. Wie bereits unter Abschnitt 4.4 beschrieben, wird das Llama-Modell nicht mittels Reinforcement Learning auf die Beantwortung von Fragen trainiert. Aus diesem Grund wurden Anweisung, Frage und Kontext in einem Textblock zusammengefasst. Die Struktur ist in Abb. 5.1 dargestellt. Eine Referenz zum Evaluationsdatensatz befindet sich im Anhang.

Die generierten Fragen wurden vor der Verwendung in Dateien in die drei Fragetypen „single“, „multi“ und „transfer“ unterteilt. Zusätzlich wurden alle Fragen in einen zweiten Datensatz kopiert, wobei die „transformed“ Fragen mit kleineren Grammatik- und Rechtschreibfehlern versehen wurden. Dieser zweite Datensatz wird in gleicher Weise im folgenden Schritt unter Abschnitt 5.4 zur Bewertung des Kriteriums „Robustheit“ in der Auswertung verwendet.

5.4 BEWERTUNG DER GENERIERTEN ANTWORTEN

Die Bewertung der generierten Antworten erfolgte manuell. Die von den Modellen generierten Ausgaben wurden wie in Abschnitt 5.3.1 strukturiert. Drei zusätzliche Eigenschaften wurden hinzugefügt.

- **generated**: Stellt die generierte Antwort des Modells dar
- **type**: Gibt den Typ der Frage an

- **true_input**: Enthält die tatsächlich verwendete Eingabe

Bei der Auswertung wurde zunächst beurteilt, ob die generierte Antwort einen inhaltlichen Bezug zur Frage hat. War dies nicht der Fall, wurde diese Frage als „unbeantwortet“ gewertet und enthielt auch keine Antwort. Wenn die Antwort einen inhaltlichen Bezug zur Frage hatte, wurde die Antwort hinsichtlich ihrer Korrektheit bewertet. Dazu gab die Bewertung eine Anzahl richtiger Antworten in der generierten Antwort an, die zwischen 0 und der erwarteten Anzahl richtiger Antworten liegt. Schließlich wurde die Anzahl der gegebenen Antworten in der generierten Antwort angegeben.

Bei der Evaluation stellte sich schnell heraus, dass die Beurteilung der Llama-Modelle schwieriger ist als die des GPT4-Modells. Der Grund dafür ist das fehlende Training auf ein Stoppzeichen. Die Llama-Modelle enthalten kein Finetuning für die Beantwortung der Fragen und erzeugen eine vorgegebene Anzahl von Tokens. Diese Anzahl wurde auf 512 gesetzt. In den meisten Fällen wurde die Frage jedoch mit einer kürzeren Länge beantwortet. Die verbleibenden Tokens enthielten Wiederholungen bereits generierter Tokens, Formatierungsfehler (z. B. nicht vorhandene Überschriften), den Beginn neuer Themen oder die Imitation neuer Fragen. Da die Ursache in einem fehlenden Finetuning und nicht in der inhaltlichen Wiedergabe des Modells lag, wurde die Antwort nur bis zu diesen Stellen gewertet. Keine der generierten Antworten benötigte eine Länge von mehr als 512 Tokens, weshalb eine Erweiterung hier nicht weiter verfolgt wurde. Die Chat-Version des Modells Llama 2 enthält ein solches Finetuning und würde dieses Problem beheben. Auf den Einsatz der Chat-Version wurde in dieser Arbeit jedoch verzichtet, da das mit Hilfe des Finetunings zusätzlich erlernte Wissen durch das hier durchgeführte Continual Pretraining wieder verlernt werden würde.

Sowohl die Antwortdateien der Modelle als auch die Bewertungsdateien folgen dem gleichen Schema wie die Eingabedatei aus Abschnitt 5.3.1. Durch die Einführung der Eigenschaft „type“ ist eine Unterteilung der Dateien in Fragetypen nicht mehr notwendig, daher gibt es 4 Dateien pro Modell. Zwei Dateien, die die generierten Antworten auf die Fragen enthalten (für jeden Datensatz, wobei der zweite Datensatz die selben Fragen mit Rechtschreibfehlern repräsentiert) und zwei Dateien, die die Bewertungen der generierten Antworten enthalten. Die Bewertungen wurden mit 3 zusätzlichen Eigenschaften versehen:

- **answered**: Zeigt an, ob die Frage richtig, falsch oder gar nicht beantwortet wurde
- **points**: Gibt die Anzahl der richtigen Antworten in der generierten Antwort an

- **total_answers**: Zeigt die Anzahl der gegebenen Antworten in der generierten Antwort an

5.5 EVALUATION DER MODELLE

Die Evaluierung der Modelle wurde für jedes ausgewählte Kriterium aus Abschnitt 4.5 durchgeführt. Für jedes Kriterium wurde ein Skript geschrieben, welcher die Bewertungsdateien der Modelle einliest und die Ergebnisse grafisch darstellt. Das Kriterium „Determinismus“ wurde nicht bewertet. Die Autoren des Artikels Omar u. a. (2023) beschreiben ein inhärentes, nicht-deterministisches Verhalten bei der Verwendung von ChatGPT. Dieses ist unter anderem auf den eingebauten Parameter „temperature“ bei der Textgenerierung zurückzuführen. Die Temperatur beschreibt die Varianz der Ausgabewerte. Je höher die Temperatur, desto größer ist die Varianz der Ausgabevektoren. Eine höhere Varianz führt zu einer größeren Vielfalt des generierten Textes und zu weniger sich wiederholenden Textpassagen. Dieser Temperaturwert wurde für die Generierung auf $0,9$ gesetzt. Das beschriebene nicht-deterministische Verhalten konnte bei der Textgenerierung mit den Llama-Modellen nicht beobachtet werden. Mehrfache identische Eingaben führten bei allen Modellen zu identischen Ausgaben. Da sich somit ein konstanter Wert von $1,0$ für den Determinismus ergibt, wurde dieser hier nicht weiter ausgewertet. Alle Kriterien wurden zusätzlich für jeden Fragetyp und jede Fragequelle ausgewertet. Die Ergebnisse dieser Auswertung sind in Kapitel 6 beschrieben.

5.5.1 *Kriterium: Korrektheit*

Für die Analyse der Korrektheit wurden alle Bewertungen der Fragen pro Modell, Fragetyp und Fragequelle gezählt. Mit diesen Werten wurde dann der MakroF1-Wert nach den Formeln aus Abschnitt 4.5 berechnet. Die Bewertungen ergeben sich aus dem Vergleich der Musterlösung mit den generierten Antworten. Die Musterlösungen sind im Buch angegeben oder wurden von den Autoren der Klausuren erstellt. Die Zählung der richtigen Antworten erfolgte somit weitestgehend objektiv.

5.5.2 *Kriterium: Erklärbarkeit*

Eine Antwort eines Modells auf eine Frage gilt als erklärt, wenn sie inhaltlich mit der Frage zusammenhängt und eine Erklärung oder nähere Beschreibung enthält. Alle Ausgaben der Modelle wurden erneut manuell auf das Vorhandensein einer Erklärung im Rahmen dieses Kriteriums bewertet. Ein Beispiel für eine Antwort mit und ohne Erklärung ist in Tabelle 5.10 dargestellt.

Erklärung	Frage	Generierte Antwort
enthalten	Is openEHR a standard?	OpenEHR (Sect. 3.7.2.2) is an open standard specification which focuses on the representation of clinical concepts in detailed clinical models. It has been maintained by the openEHR Foundation since the early 20s and supports the creation of clinical models in the form of a centralized database. [...]
nicht enthalten	What types of integrity exist in the context of health information systems?	- data integrity, - semantic integration, - user interface integration, - context integration, - feature integration, - process integration.[...]

Tabelle 5.10: Ein Beispiel für Antworten mit Erklärung und ohne Erklärung

5.5.3 *Kriterium: Fragenverständnis*

Das Kriterium Fragenverständnis bezieht sich auf die Fähigkeit eines Modells, die gestellten Fragen zu verstehen. Eine Frage gilt als verstanden, wenn sich die generierte Antwort inhaltlich auf die Frage bezieht und darüber hinaus alle Aspekte der Frage adressiert. Die Bewertung des Fragenverständnisses erfolgte analog zur Bewertung der Erklärbarkeit.

5.5.4 *Kriterium: Robustheit*

Die Robustheit eines Modells beschreibt die Fähigkeit, auch bei fehlerhaften Eingaben eine korrekte Antwort zu generieren. Zu diesem Zweck wurde, wie bereits erwähnt, während der Generierung der Ausgaben ein zusätzlicher Datensatz ausgewertet, der die selben Fragen wie der ursprüngliche Datensatz enthielt, jedoch mit kleineren Rechtschreib- und Grammatikfehlern versehen war. Da sich die Robustheitsbewertung auf richtig beantwortete Fragen bezieht, die auch bei falscher Eingabe richtig beantwortet werden, wurde dieser Datensatz für jedes Modell reduziert und enthielt nur Fragen, die das Modell zuvor mit mindestens einer richtigen Antwort beantwortet hatte. Die Berechnung der Ergebnisse erfolgt analog zur Berechnung des Kriteriums Korrektheit, wobei hier nicht die Anzahl der im Ro-

bustheitsdatensatz enthaltenen Fragen, sondern die Anzahl aller vorhandenen Fragen für die Berechnung des MakroF1-Wertes verwendet wurde.

6 ERGEBNISSE

Ausführliche Ergebnisse und die verwendeten Datensätze sind unter `https://doi.org/10.5281/zenodo.8363501` verfügbar.

In dieser Arbeit wurde die Konzeption und Implementierung eines Continual Pretrainings von Llama-Modellen beschrieben. Diese Modelle wurden anschließend nach den in Abschnitt 4.5 genannten Kriterien Korrektheit, Erklärbarkeit, Fragenverständnis und Robustheit verglichen. Die Durchführung dieses Vergleichs ist in Abschnitt 5.5 beschrieben. In diesem Kapitel werden die Ergebnisse dargestellt und analysiert, wobei auch hier eine Unterteilung nach den Kriterien erfolgt. Die zu vergleichenden Modelle sind in Tabelle 6.1 aufgeführt.

6.1 ANALYSE KORREKTHEIT

6.1.1 *Vergleich totaler Zahlen*

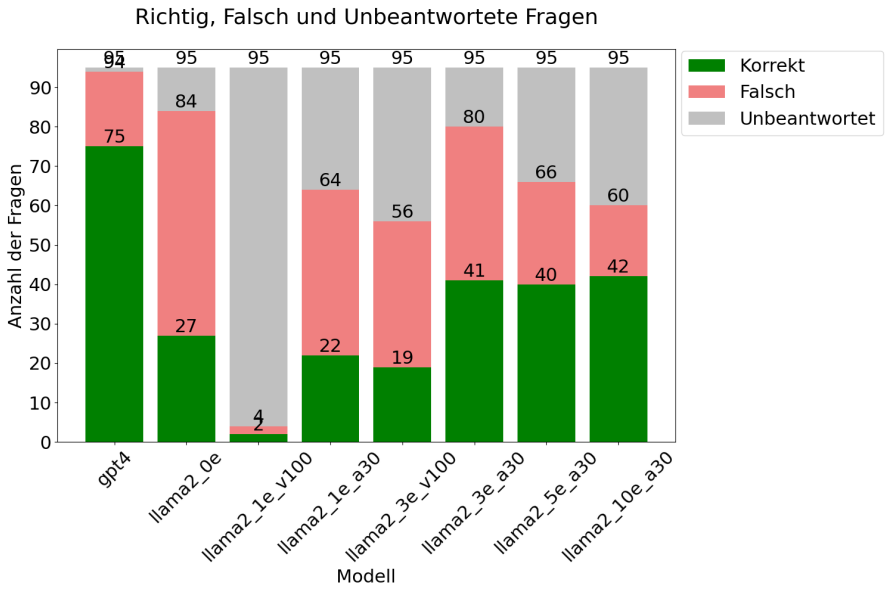

Abbildung 6.1: Vergleich evaluierter Modelle und ihren totalen Leistungen bei der Beantwortung von Fragen.

Abb. 6.1 zeigt die Verteilung der richtigen, falschen und unbeantworteten Fragen der evaluierten Modelle. Allen Modellen wurden dieselben 95 Fragen mit identischem Kontext vorgelegt. Diese absoluten Zahlen zeigen eine grobe Tendenz der Ergebnisse, spiegeln aber nicht die tatsächlichen Leistungen wider. Eine Frage gilt als richtig beantwortet, wenn mindestens eine richtige Antwort gegeben wurde. Dies führt zu irreführend guten Ergebnissen. Dennoch ist deutlich zu erkennen,

Modell	Epochen	Grafikkarten	Bezeichnung
GPT4	-	-	`gpt4`
Llama 2 7B	0	Nvidia Tesla A100	`llama2_0e`
	1	Nvidia Tesla V100	`llama2_1e_v100`
	1	Nvidia Tesla A30	`llama2_1e_a30`
	3	Nvidia Tesla V100	`llama2_3e_v100`
	3	Nvidia Tesla A30	`llama2_3e_a30`
	5	Nvidia Tesla A30	`llama2_5e_a30`
	10	Nvidia Tesla A30	`llama2_10e_a30`

Tabelle 6.1: Evaluierte Modelle, deren Anzahl an Epochen, genutzte Grafikkarten und Bezeichnungen in den Grafiken.

dass GPT4 den Llama-Modellen deutlich überlegen ist. Das untrainierte Llama-Modell beantwortete 28 % der Fragen mit mindestens einer richtigen Antwort, konnte aber auch einen Großteil der Fragen beantworten und lieferte bei 11 % keine Antwort.

Das `llama2_1e_v100`-Modell liefert mit 2 % die wenigsten korrekten Antworten und zeigt ein generelles Verlernen grundlegender sprachlicher Fähigkeiten. Im Gegensatz dazu zeigt das `llama2_1e_a30`-Modell eine vergleichbare, wenn auch etwas schlechtere Leistung als das untrainierte Modell.

Mit dem `llama2_3e_v100`-Modell konnte eine signifikante Leistungssteigerung im Vergleich zu einer Epoche erzielt werden, die Gesamtleistung ist jedoch dauerhaft schlechter als beim untrainierten Modell. Das `llama2_3e_a30`-Modell hingegen übertraf erstmals die Leistung des untrainierten Modells und beantwortete 43 % der Fragen mit mindestens einer richtigen Antwort.

Modelle mit höherer Epoche wurden ausschließlich auf A30-Grafikkarten trainiert und zeigten keine weitere Leistungssteigerung, beantworteten aber kontinuierlich weniger Fragen. Hier zeigt sich eine Präferenz für keine Antwort gegenüber einer falschen Antwort.

6.1.2 *Stärken und Schwächen der Modelle*

Neben der Gesamtleistung der Modelle wurde auch eine Unterteilung nach Fragetypen und Fragequellen vorgenommen. Ähnliche Grafiken wie in Abb. 6.1 sind in den ergänzenden Materialien zu dieser Arbeit enthalten.

Fragetyp	Anzahl der Fragen	Bezeichnung
Singulär-Fakt	34	„single“
Multi-Fakten	38	„multi“
Transfer	23	„transfer“

Tabelle 6.2: Fragetypen die im Evaluierungsdatensatz enthalten sind.

Fragequelle	Anzahl der Fragen	Bezeichnung
Health Information Systems von Winter u. a. (2023)	33	„book“
Mündliche Klausurfragen aus dem Modul „Architektur von Informationssystemen im Gesundheitswesen“ vom Jahr 2021	22	„A_2021“
Schriftliche Klausurfragen aus dem Modul „Informationssysteme in medizinischer Versorgung und Forschung“ vom Jahr 2022	9	„IS_2022_07_18“
Schriftliche Klausurfragen aus der Nachholklausur des Moduls „Informationssysteme in medizinischer Versorgung und Forschung“ vom Jahr 2022	31	„IS_2022_09_27“

Tabelle 6.3: Fragequellen die im Evaluierungsdatensatz enthalten sind.

Die verschiedenen Fragetypen sind in Tabelle 6.2 aufgelistet. In den folgenden Grafiken werden die entsprechenden Bezeichnungen zur Beschriftung der Achsen verwendet.

Unterschiedliche Fragequellen sind in Tabelle 6.3 aufgelistet. Da die Quelle „IS_2022_07_18“ nur 9 Fragen enthält, ist eine Bewertung der Leistung in dieser Rubrik ungenau und anfällig für kleine Änderungen. Aus diesem Grund erfolgt der Vergleich der Modellleistungen unabhängig von dieser Fragequelle.

Das GPT4-Modell hat keine Präferenz für bestimmte Fragetypen, beantwortete aber die Fragen aus dem Buch mit 93 % korrekten Antworten signifikant besser. Vergleichsweise wenige Antworten wurden

bei den beiden schriftlichen Prüfungen gegeben.

Das untrainierte Llama-Modell zeigte mit 39 % richtigen Antworten eine deutliche Stärke bei Multi-Fakten-Fragen und konnte 27 % der mündlichen Klausurfragen beantworten. Die schlechtesten Kategorien ergaben sich bei den Ein-Fakt-Fragen mit 20 % und den Fragen aus dem Buch mit 24 % richtigen Antworten.

Während das `llama2_1e_v100`-Modell mit nur 2 richtig beantworteten Fragen grundsätzlich schlecht abschnitt, zeigte das `llama2_1e_a30`-Modell mit 36 % eine deutlich bessere Leistung bei Fragen mit mehreren Fakten und bei der mündlichen Prüfung mit 40 % richtig beantworteten Fragen. Dagegen verlor dieses Modell deutlich bei Ein-Fakten-Fragen mit 5 % und Klausurfragen mit nur 12 % richtigen Antworten.

Modelle, die auf 3 Epochen trainiert wurden, zeigen einen deutlichen Leistungssprung gegenüber einer Epoche. Sowohl das `llama2_3e_v100`-Modell als auch das `llama2_3e_a30`-Modell verbessern ihre Leistung in den zuvor als schwach identifizierten Fragetypen und Fragequellen. Im Falle des `llama2_3e_a30`-Modells bedeutet dies eine Verdoppelung der richtig beantworteten Transferfragen sowie eine Verdoppelung der richtig beantworteten Fragen aus dem Buch. Diese Leistungssteigerung zeigt sich auch in den MakroF1-Werten.

Epoche 5 und Epoche 10 enthalten nur Modelle, die mit Nvidia A30 Grafikkarten trainiert wurden. Die gezeigten Leistungen unterscheiden sich kaum von denen des `llama2_3e_a30`-Modells und weisen nur minimale Leistungsschwankungen auf. Allerdings verschiebt sich die Verteilung der richtig beantworteten Fragen hin zu Fragen aus dem Buch. Die anderen Fragequellen werden von den Modellen mit zunehmender Epoche immer schlechter beantwortet, während die Fragen aus dem Buch immer besser beantwortet werden. Diese Verschiebung deutet auf eine Überanpassung hin, da die Fragen aus dem Buch auch in den Trainingsdaten enthalten sind.

6.1.3 *Verbesserungen durch Training*

Während der Trainingsphase wurden die Modelle mit Hilfe eines Validierungsdatensatzes evaluiert. Diese zusätzliche Evaluierung berechnete den Fehlerwert des Modells auf einem gegebenen Datensatz. Abb. 6.2 zeigt die Fehlerwerte der Trainingsdaten und Abb. 6.3 die Fehlerwerte der Validierungsdaten für jedes Modell. Fehlerwerte beschreiben nicht konkret die Leistung eines Modells, sondern sind ein Indikator für die Verbesserung der Leistung und den Beginn der Überanpassung. Ein abnehmender Fehlerwert des Trainingsdatensatzes

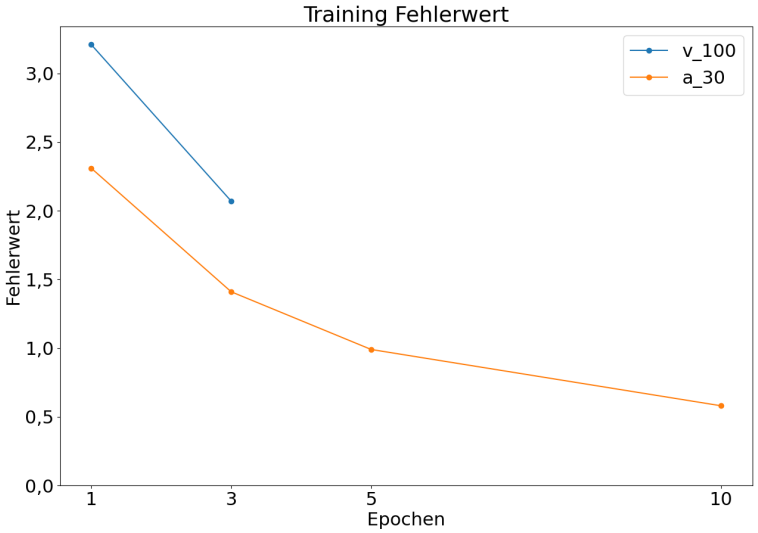

Abbildung 6.2: Vergleich evaluierter Modelle und ihrer Fehlerwerte des Trainingsdatensatzes.

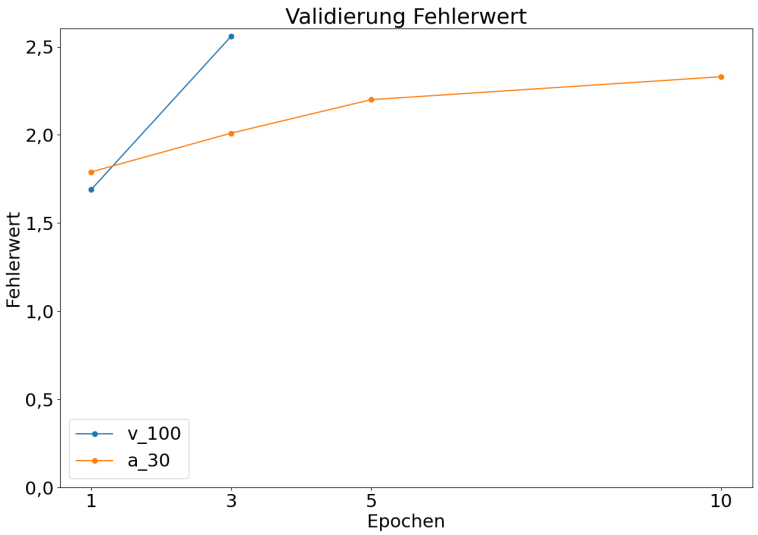

Abbildung 6.3: Vergleich evaluierter Modelle und ihrer Fehlerwerte des Validierungsdatensatzes.

bedeutet, dass das Modell den Trainingsdatensatz besser nachahmen kann, während ein abnehmender Fehlerwert des Validierungsdatensatzes zeigt, dass das Modell diese Nachahmung generalisierend auf ungesehenen Text anwenden kann.

Wenn der Fehlerwert des Validierungsdatensatzes zu steigen beginnt, deutet dies auf eine Überanpassung hin. Das Modell verliert dann die Fähigkeit, ungesehenen Text zu imitieren und beginnt stattdessen, den Trainingsdatensatz auswendig zu lernen. Dieses Phänomen ist bei beiden Modelltypen bereits ab Epoche 3 zu beobachten und setzt sich konstant bis Epoche 10 fort.

Für die Beantwortung der Fragen ist das Auswendiglernen jedoch in gewissem Maße vorteilhaft, da Definitionen von Begriffen oder Aufzählungen von Fakten vom Modell besser zitiert werden können. Dies spiegelt sich auch in den Ergebnissen wider. Modelle der Epoche 3 können Fragen aus allen Fragequellen und Fragetypen besser beantworten, da hier die Zitierfähigkeit des Modells erhöht wird. Ein zu hoher Grad an Überanpassung führt jedoch zu anderen Problemen. Treten unbekannte Formulierungen wie z.B. Rechtschreibfehler auf, kann das Modell diese nicht mehr korrekt beantworten. Diese Problematik nimmt mit steigender Epoche zu, weshalb Modelle ab Epoche 5 in ihrer Leistungsfähigkeit nachlassen.

6.1.4 *Vergleich MakroF1*

Abb. 6.4 zeigt die MakroF1-Werte der evaluierten Modelle. Auch hier sind die Tendenzen aus Abschnitt 6.1.2 erkennbar. GPT4 erreicht einen MakroF1-Wert von 0,7 und ist damit deutlich besser als die Llama-Modelle. Während die auf 1 Epoche trainierten Modelle noch unter der Leistung des untrainierten Llama-Modells liegen, verdoppeln sich die MakroF1-Werte der Modelle ab 3 Epochen. Danach bleiben die MakroF1-Werte der Modelle mit 5 und 10 Epochen konstant.

Der MakroF1-Wert hat einen Wertebereich von 0 bis 1 und repräsentiert die tatsächliche Leistung des Modells besser. Modelle, die Fragen richtig beantwortet haben, aber auch falsche oder unvollständige Antworten gegeben haben, erhalten hier einen niedrigeren F1-Wert. Ein Modell mit einem MakroF1-Wert von 0 beantwortet keine Frage richtig, während ein Modell mit einem Wert von 1 alle Fragen richtig und vollständig beantwortet. So beantwortet das GPT4-Modell im Durchschnitt eine Frage zu 70 % richtig, während das `llama2_3e_a30`-Modell im Durchschnitt eine Frage zu 30 % richtig beantwortet.

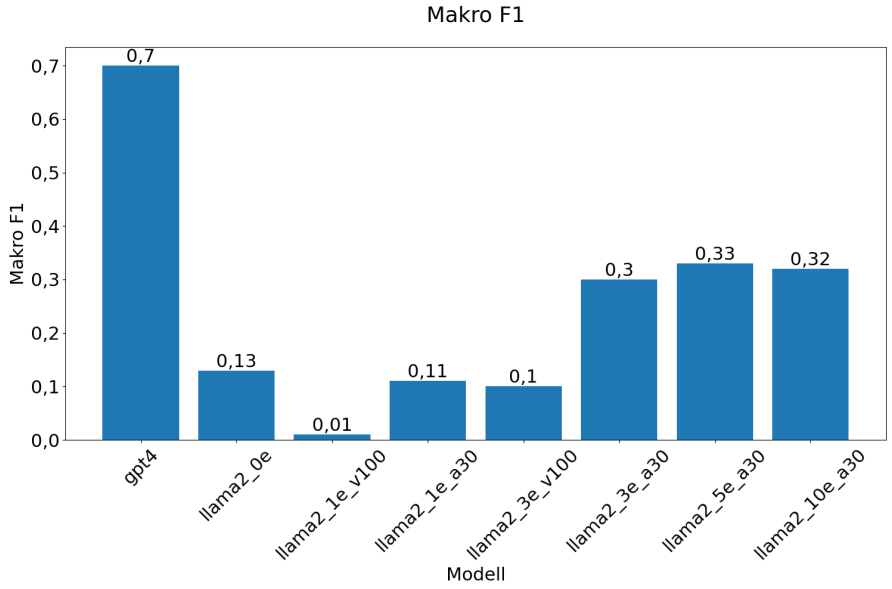

Abbildung 6.4: Vergleich evaluierter Modelle und Makro F1 Werten bei der Beantwortung von Fragen.

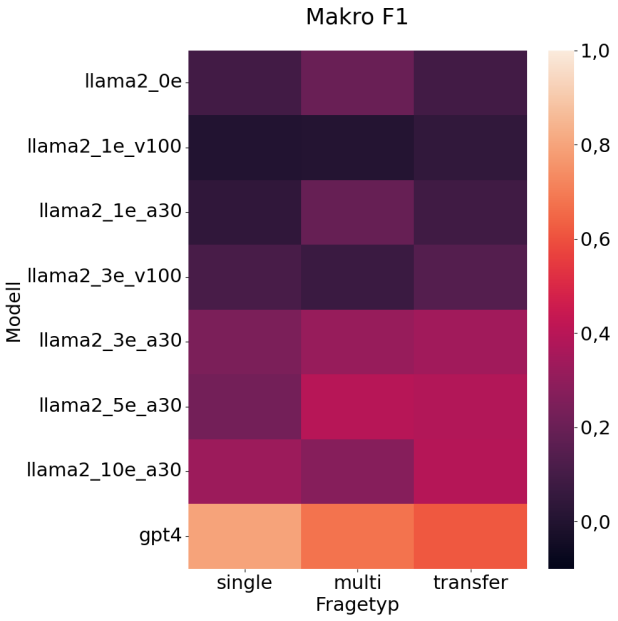

Abbildung 6.5: Heatmap der MakroF1-Werte der evaluierten Modelle nach Fragetypen.

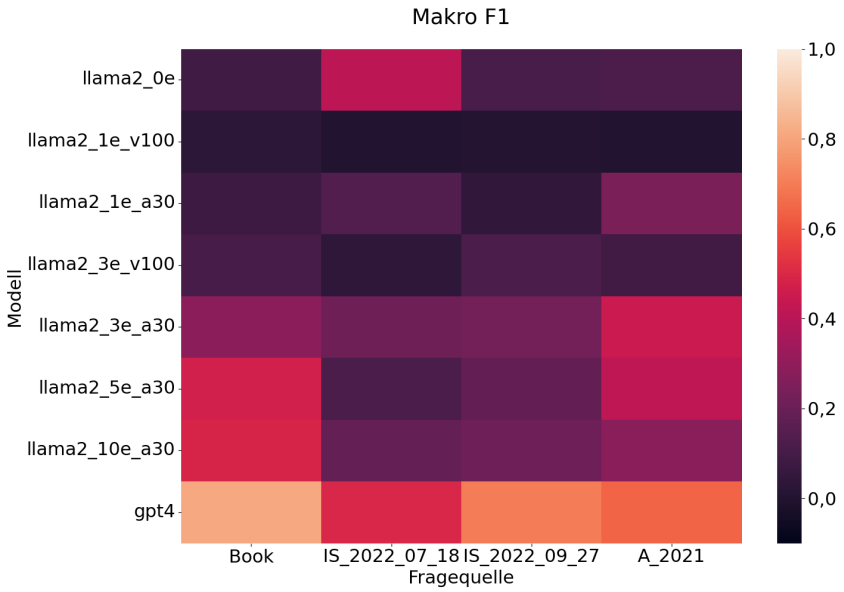

Abbildung 6.6: Heatmap der MakroF1-Werte der evaluierten Modelle aufgeteilt nach Fragequellen.

Eine genauere Aufschlüsselung der MakroF1-Werte nach Fragetypen ist in Abb. 6.5 zu sehen. Hier sieht man die Leistungssteigerung in allen Fragetypen ab 3 Epochen. Nicht intuitiv werden Transferfragen von den Modellen besser beantwortet als Einzelfragen, obwohl letztere für den Menschen deutlich einfacher sind. Der Grund dafür könnte in der günstigen Grundstruktur der Modelle liegen. Durch die inhärente Architektur mit Hilfe einer Aufmerksamkeitsmaske sind Transformermodelle darauf spezialisiert, Texte und Token miteinander zu korrelieren und könnten daher verschiedene Textstellen aus dem Buch und Fakten besser miteinander verknüpfen. Diese Beobachtung kann jedoch nicht auf das GPT4-Modell übertragen werden. Da der technische Hintergrund des GPT4-Modells nicht bekannt ist, kann diese Beobachtung nicht weiter begründet werden. Das GPT4-Modell wurde nach normalem Training durch umfangreiches Finetuning mit menschlichem Feedback an die Beantwortung von Fragen angepasst, wodurch die Leistung bei der Beantwortung von einfachen Einzelfragen hier durchaus gesteigert werden könnte.

Abb. 6.6 zeigt die MakroF1-Werte der Modelle unterteilt nach Fragequellen. Die Entwicklung der Modelle von einem untrainierten über einen trainierten zu einem übertrainierten Zustand ist hier gut zu erkennen. Da die Fragequelle „IS_2022_07_18“ nur aus 9 Fragen besteht, ist eine generelle Aussage hier nicht möglich. Kleinere Schwankungen, wie z.B. die richtige Beantwortung einer Frage, führen zu großen Ausschlägen, die fälschlicherweise als sehr gute Leistung interpretiert werden könnten.

Bis zur Epoche 3 ist eine stetige Leistungssteigerung in allen Fragequellen zu beobachten, wobei Fragen aus der Quelle „A_2021“ am meisten profitieren. Dies ist zu erwarten, da das zugrundeliegende Modul inhaltlich sehr eng mit dem Buch *Health Information Systems* zusammenhängt. Ab Epoche 5 ist eine Verschiebung in Richtung der Fragequelle „book“ zu erkennen, was auf ein Überanpassen hindeutet. Ab diesem Zeitpunkt können die Modelle Fragen aus dem Buch besser beantworten, indem sie die darin enthaltenen Antworten zitieren, während andere Formulierungen desselben Wissens zu Unverständnis führen. Dennoch ist auch diese Leistungssteigerung ein guter Fortschritt und zeigt die Stärken der Informationsextraktion von Transformermodellen aus Text. Die im Buch gestellten Fragen werden nicht unmittelbar nach der Frage beantwortet, sondern erst am Ende des Buches. Das bedeutet, dass die Transformermodelle hier trotz der großen Distanz eine klare Verbindung zwischen Frage und Antwort herstellen konnten.

6.1.5 *Zusammenfassung*

Die MakroF1-Werte bestätigen die zuvor gemachten Aussagen über die Gesamtanzahl der Fragen zur Leistungsfähigkeit der Modelle. GPT4 erreicht hier mit 0,7 den höchsten MakroF1-Wert, vergleichbar mit den Ergebnissen aus OpenAI (2023) zum Thema „Medical Knowledge“. Das Llama 2 7B startet im untrainierten Zustand deutlich unter dem normalen Leistungsniveau. Seine Leistung kann durch ein Continual Pretraining ab Epoche 3 verdoppelt werden. Eine Epoche, d.h. das einmalige Lesen des Buches, scheint nicht auszureichen, um eine bessere Leistung zu erzielen. Dies liegt auch an der Größe des Trainingsdatensatzes. Da dieser relativ klein ist, kann das Modell seine Gewichte nicht schnell genug anpassen, um die darin enthaltene Information zu reproduzieren. Eine höhere Lernrate würde eine schnellere Anpassung des Modells an den gesehenen Text verbessern, aber die Gefahr erhöhen, dass sich das Modell zu sehr an Formulierungen und Formatierungen anpasst und dadurch unbrauchbar wird.

Ab Epoche 5 verliert das Modell an Leistung, obwohl die ermittelten MakroF1-Werte konstant bleiben. Ab diesem Zeitpunkt beginnt das Modell die Verallgemeinerung zu verlernen und kann nur noch Fragen aus dem Buch mit zitierten Antworten richtig beantworten. Aus diesem Grund scheint das Modell seine beste Leistung ab Epoche 3 zu erreichen. Hier erreicht das Modell besonders gute MakroF1 durch eine übermäßig gute Leistung bei Fragen aus der Fragequelle „A_2021“.

GPT4 ist durch seine Größe und die dahinter stehende Trainingsmenge unschlagbar. Es ist zu erwarten, dass dieses Modell die Leistung eines Modells mit 7 Milliarden Parametern übertrifft. Die Leis-

Frage	Definieren Sie den Begriff „Krankenhausinformationssystem“.
Übersetzt	Define the term “hospital information system”.
Erwartete Antwort	A Hospital Information system is the sociotechnical subsystem of hospitals. It comprises all data, information, and knowledge processing as well as the associated human or technical actors in their respective data, information, and knowledge processing roles.
Generierte Antwort	### 2.2.2.2.2.2.1.1.1.1.1.2.1.1.2.1.2.1.2.1.2.1.2. [...]

Tabelle 6.4: Beispiel für Formatierungsfehler in der Ausgabe des Llama 2 7B Modells trainiert auf 1 Epoche

tungssteigerung des Llama-Modells zeigt jedoch einen Trend, der durch größere Modelle noch weiter gesteigert werden könnte. Zum jetzigen Zeitpunkt sind die trainierten Modelle aus dieser Arbeit jedoch nicht verwendbar. Insbesondere in Epoche 1 enthalten die Antworten grundlegende Fehler in der Ausgabe durch Imitation von Formatierungen (siehe Tabelle 6.4). Weiter trainierte Modelle zeigen eine Präferenz für keine Antwort gegenüber einer falschen Antwort, was für die Anwendung des Modells durchaus vorteilhaft sein kann. Allerdings ist diese Tendenz auf eine Überanpassung an den Trainingsdatensatz zurückzuführen.

Modelle, die mit V100-Grafikkarten trainiert wurden, zeigen eine sehr starke Verschlechterung der Leistung. Dies ist auf das notwendige Training im FP16 Datenformat gegenüber dem ursprünglich verwendeten BF16 Datenformat zurückzuführen. Wie auch von Huggingface beschrieben, sollten vortrainierte Modelle weiterhin im vorgesehenen Datenformat trainiert werden. Diese Beobachtung wird auch durch die Ergebnisse dieser Arbeit bestätigt.

6.2 ANALYSE ERKLÄRBARKEIT

Die hier evaluierten Modelle weisen eine hohe Fähigkeit auf, Erklärungen zu den generierten Antworten zu liefern. Insbesondere GPT4 ist in der Lage, zu 97 % richtig beantworteten Fragen eine Erklärung zu liefern. Abb. 6.7 zeigt hier die Gesamtzahl der Antworten, die Erklärungen enthalten. Dabei wurden nur Fragen berücksichtigt, die das jeweilige Modell zuvor mit mindestens einer richtigen Antwort

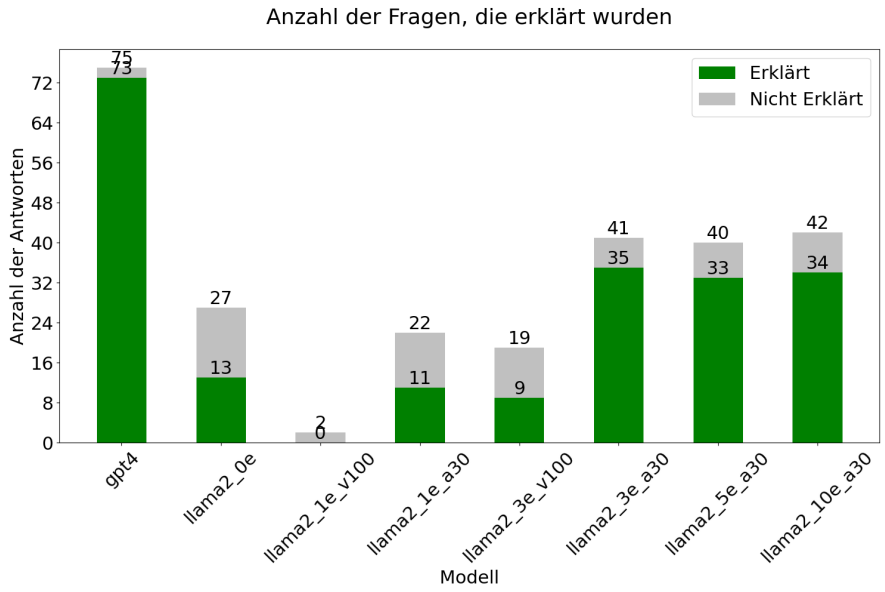

Abbildung 6.7: Vergleich der evaluierten Modelle und der Anzahl der Antworten mit Erklärungen.

beantwortet hat. Das untrainierte Modell sowie die trainierten Modelle einer Epoche können hier nur für die Hälfte der Fragen eine Erklärung liefern. Ab Epoche 3 steigt diese Leistung ebenso deutlich auf 85 %.

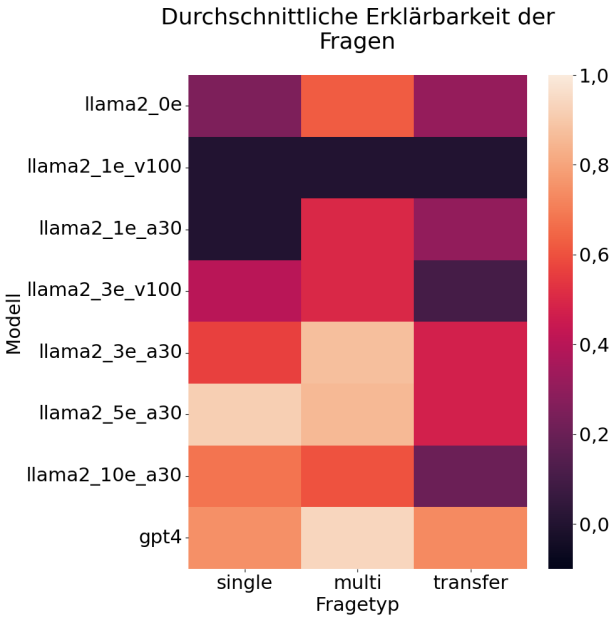

Abbildung 6.8: Heatmap der Erklärbarkeit der evaluierten Modelle nach Fragetypen.

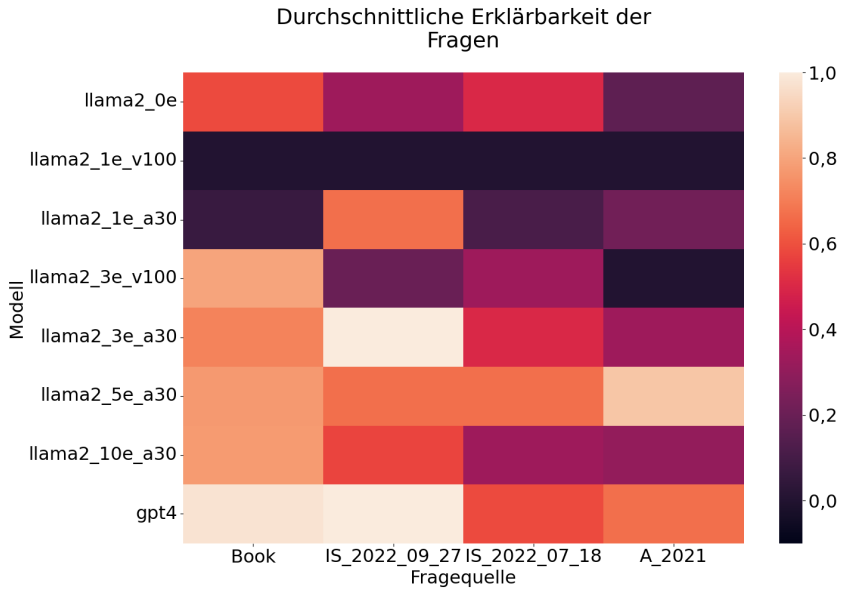

Abbildung 6.9: Heatmap der Erklärbarkeit der evaluierten Modelle unterteilt nach Fragequellen.

Abb. 6.8 und Abb. 6.9 zeigen die Unterteilung dieser Leistung nach Fragetypen und Fragequellen. Hier ist zunächst kein klarer Trend der Verbesserung in diesen Unterteilungen zu erkennen. Insbesondere Fragen des Fragetyps „Transfer“ enthalten wenig Erklärungen, während Fragen des Fragetyps „Multi“ am meisten Erklärungen enthalten. Die zuvor gut erscheinenden Ergebnisse bei Fragen des Fragetyps „A_2021“ zeigen hier, dass die generierten Antworten zwar korrekt sind, aber überwiegend keine Erklärungen enthalten. Hervorzuheben ist die grundsätzlich gute Leistung von GPT4, die durch das eingesetzte Finetuning mit menschlicher Bewertung (engl. „Human Reinforcement Learning“) begründet werden können. Mit dieser Trainingsmethode könnte auch das Llama-Modell belohnt werden, wenn es Erklärungen für seine Antworten gibt.

6.3 ANALYSE FRAGENVERSTÄNDNIS

Das Kriterium Fragenverständnis ist nicht äquivalent zur Anzahl der beantworteten Fragen des Kriteriums Korrektheit. Eine Frage gilt nicht als verstanden, wenn sie beantwortet wurde, da falsche Antworten zwar falsche Fakten enthalten können, die Frage aber nicht vollständig beantwortet wurde und daher als nicht verstanden gezählt wird. Ebenso kann eine unbeantwortete Frage als verstanden gezählt werden. Beispielsweise generierte GPT4 in einigen Fällen eine Antwort, die erklärte, warum das Modell die Frage nicht beantworten konnte, die Frage aber eindeutig verstanden hatte.

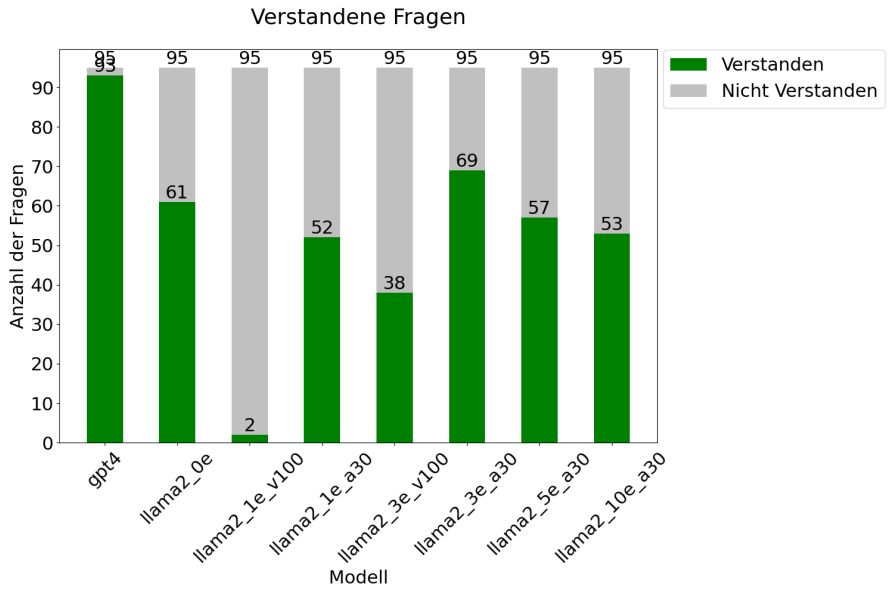

Abbildung 6.10: Vergleich der evaluierten Modelle und der Anzahl der verstandenen Fragen.

Abb. 6.10 zeigt die Gesamtzahl der verstandenen Fragen der evaluierten Modelle. Im Gegensatz zum Kriterium Erklärbarkeit wurden hier alle vorhandenen Fragen auf ihr Verständnis hin überprüft. Auch hier schneidet GPT4 mit 97 % verstandenen Fragen deutlich besser ab als die Llama-Modelle. Lediglich das `llama2_3e_a30`-Modell erreicht mit 72 % verstandenen Fragen eine etwas bessere Leistung als das untrainierte Modell mit 64 %. Insbesondere Modelle mit höherer Epoche verlieren die Fähigkeit, Fragen zu verstehen. Diese Beobachtung bestätigt somit auch die steigende Anzahl unbeantworteter Fragen aus dem Kriterium Korrektheit, da immer mehr Fragen nicht verstanden werden. Grund hierfür ist die zunehmende Überanpassung der Modelle, die es den Modellen erschwert, andere Formulierungen gleichen Wissens zu verstehen.

Abb. 6.11 und Abb. 6.12 zeigen die Unterteilung dieser Statistik nach Fragetyp und Fragequelle. Auch hier ist ein deutlicher Leistungsabfall ab Epoche 5 zu erkennen, insbesondere beim Fragetyp „transfer“. Das `llama2_3e_a30`-Modell zeigt hier die beste Leistung aller Llama-Modelle, obwohl es in der Kategorie „multi“ schlechter als das untrainierte Modell abschneidet. Die Unterteilung in Fragequellen bestätigt diese Beobachtungen, wobei die Quelle „IS_2022_09_27“ besonders von der Überanpassung der Modelle ab Epoche 5 betroffen ist. Die Fragen dieser Quelle scheinen grundsätzlich andere Formulierungen als im Buch zu verwenden und müssen daher vermehrt mit generalisiertem Wissen beantwortet werden.

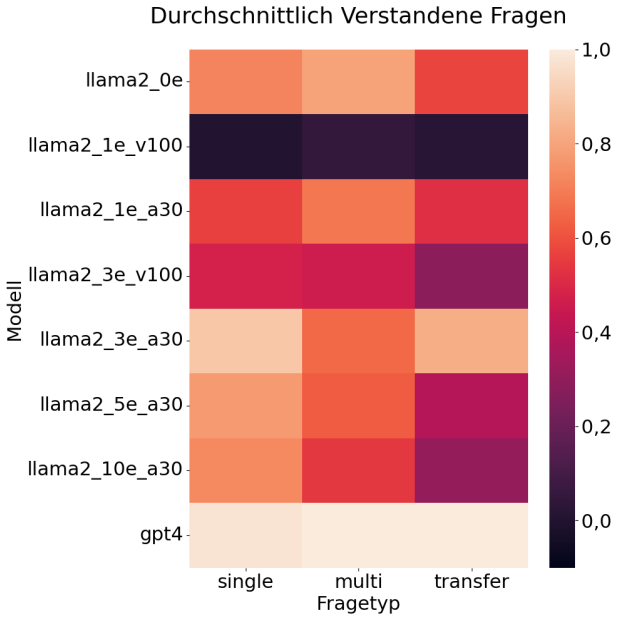

Abbildung 6.11: Heatmap des Fragenverständnisses der evaluierten Modelle nach Fragetypen.

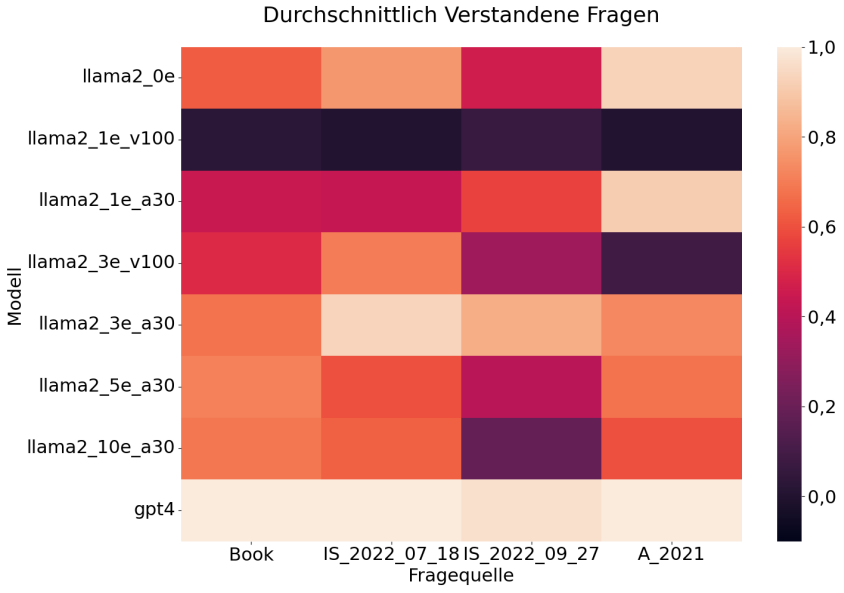

Abbildung 6.12: Heatmap des Fragenverständnisses der evaluierten Modelle unterteilt nach Fragequellen.

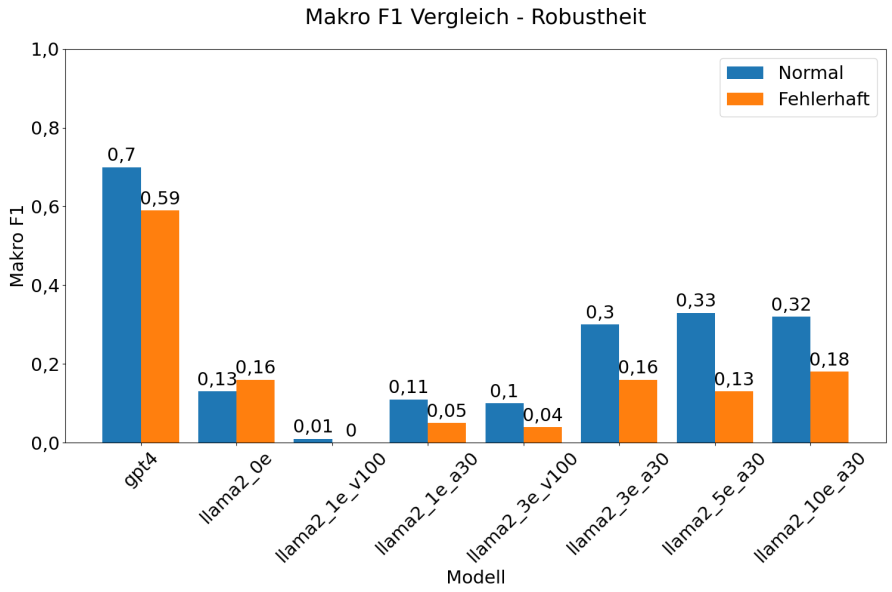

Abbildung 6.13: Vergleich der evaluierten Modelle und ihrer MakroF1-Werte nach Einführung von Rechtschreibfehlern.

6.4 ANALYSE ROBUSTHEIT

Abb. 6.13 zeigt die MakroF1-Werte der Modelle nach Einführung von Rechtschreibfehlern. Dazu wurden alle richtig beantworteten Fragen der Modelle mit zusätzlichen Rechtschreibfehlern versehen und erneut ausgewertet. Rechtschreibfehler in Fachausdrücken erschweren die Wiedergabe von Sachverhalten, da die Tokenisierung die Wörter anders unterteilt. Wenn Fakten direkt mit einer bestimmten Tokenisierung verknüpft sind, kann das Modell diese Fakten bei einer anderen Unterteilung nicht mehr wiedergeben. Aus diesem Grund ist eine schlechtere Leistung der Modelle zu erwarten und zeigt die Fähigkeit zur Generalisierung des gelesenen Wissens.

Auch hier schneidet das GPT4-Modell mit einem MakroF1-Wert von 0,59 gegenüber dem Ausgangswert 0,7 am besten ab. Dies entspricht einer Leistungsminimierung von 16 %. Trainierte Modelle zeigen deutlich höhere Leistungsminimierungen von etwa 50 %. Diese großen Unterschiede deuten auf einen oberflächlichen Wissenserwerb mit Fokus auf korrekt geschriebene Fachbegriffe hin. Bei wenigen Trainingsdaten können die Modelle das darin enthaltene Wissen nur grob erlernen. Hier kann die Leistung verbessert werden, indem der Trainingsdatensatz erweitert wird, um mehr Arten der Wissensformulierung zu enthalten. Dies würde die Generalisierung der Modelle und damit die Leistung in diesem Kriterium verbessern.

Das untrainierte Llama-Modell erzielt etwas bessere Ergebnisse mit eingebauten Rechtschreibfehlern. Diese Verbesserung wird aufgrund

Modell	MakroF1	Erklärbarkeit	Fragenverständnis	Robustheit (Leistungsverlust)
GPT4	**0,7**	**97,3** %	**97,9** %	15,7 %
llama2_0e	0,13	48,1 %	64,2 %	**−0,23** %
llama2_1e_v100	0,01	0 %	2,1 %	100 %
llama2_1e_a30	0,11	50 %	54,7 %	54,5 %
llama2_3e_v100	0,1	47,4 %	40 %	60 %
llama2_3e_a30	0,3	**85,4** %	**72,6** %	46,7 %
llama2_5e_a30	**0,33**	82,5 %	60 %	60,6 %
llama2_10e_a30	0,32	81 %	55,8 %	**43,8** %

Tabelle 6.5: Zusammenfassung der Evaluierungsergebnisse. Fettgedruckte Werte stellen die beste Leistung im Kriterium dar. Die Zeilen sind unterteilt in nur evaluierte Modelle und trainiert und evaluierte Modelle.

der geringen Veränderung als zufällig angesehen und bedeutet nicht, dass das Modell bessere Ergebnisse liefert, wenn Fehler enthalten sind. Besser erklärt sind hier etwa gleiche Werte. Die Llama-Modelle wurden zunächst auf sehr großen Datensätzen trainiert, die auch die Größe der Datensätze für GPT4 überstiegen. Das erlernte Wissen dahinter ist daher fundierter und kann deutlich besser verallgemeinert auf verschiedene Formulierungen angewendet werden. Durch das hier durchgeführte Continual Pretraining werden diese fundierten Wissensknoten überschrieben und durch oberflächlichen Wissenserwerb ersetzt. Dadurch kann das Modell zwar zunächst mehr Wissen abbilden, verliert aber die Fähigkeit, dieses Wissen mit unterschiedlichen Formulierungen zu verknüpfen.

6.5 ZUSAMMENFASSUNG

Tabelle 6.5 zeigt eine Zusammenfassung der Evaluationsergebnisse. Dabei werden die Modelle in die Gruppen „nur evaluiert“ und „trainiert und evaluiert“ unterteilt. Im Allgemeinen übertrifft GPT4 die Llama-Modelle in allen Kriterien mit Ausnahme des Leistungsverlustes im Kriterium Robustheit. Von den Llama-Modellen erreicht `llama2_5e_a30` die beste MakroF1-Leistung, `llama2_10e_a30` den geringsten Leistungsverlust im Kriterium Robustheit und `llama2_3e_a30` die besten Leistungen in den Kriterien Erklärbarkeit und Fragenverständnis. Unabhängig vom Robustheitskriterium scheint ein optimales Modell mit 3 bis 5 Epochen trainiert zu werden. Von den derzeit trainierten Modellen ist bei genauerer Betrachtung der Kriterien das `llama2_3e_a30`-Modell am besten geeignet, die Fragen zu beantworten, da ab 5 Epochen die Überanpassung der Modelle einen zu starken Einfluss hat.

Hinsichtlich der Erklärbarkeit und des Fragenverständnisses ist ab Epoche 3 eine deutliche Verbesserung zu erkennen. Die Leistungseinbußen durch Rechtschreibfehler zeigen jedoch, dass das Wissen nicht fundiert erworben wurde und die Modelle anfällig für andere Formulierungen sind.

Mögliche Leistungssteigerungen können durch größere Modelle erreicht werden. Die Llama2-Modelle sind zusätzlich mit 13 Milliarden und 70 Milliarden Parametern verfügbar. Größere Modelle sind besser in der Lage, komplexes Wissen zu erlernen und zu verallgemeinern. Dies kann dazu führen, dass ein längeres Training ohne Überanpassung möglich ist. Darüber hinaus hilft ein größerer Datensatz, insbesondere zur Vermeidung von Überanpassung, Informationen in verschiedenen Formulierungen darzustellen und damit die Robustheit zu verbessern. Ebenso ist eine generelle Verbesserung der Ergebnisse im Kriterium Korrektheit zu erwarten. Als letzte Möglichkeit kann ein Training mit Hilfe von Human Reinforcement Learning durchgeführt werden. Dieses ermöglicht es, das Erklären und Verstehen von Fragen zu belohnen und somit auch diese beiden Kriterien zu verbessern.

7 DISKUSSION

Die Ergebnisse dieser Arbeit zeigen, dass ein Continual Pretraining die Leistung eines Modells bei der Beantwortung von Fragen verbessern kann. Ebenso wirkt das Modell demnach unterstützend bei der Wissensbeschaffung aus Winter u. a. (2023). Dennoch erreichte das hier trainierte Modell eine maximale Leistung von 0,3 MakroF1, was nicht mit derzeit verfügbaren Modellen wie GPT4 vergleichbar ist. Um diese Leistung zu erreichen, werden in Kapitel 8 einige Ansätze vorgestellt, die in zukünftigen Arbeiten untersucht werden können.

7.1 GRENZEN DER MODELLE

Die hier vorgestellten Modelle stellen eine Momentaufnahme in einer bestimmten Umgebung dar, können aber keine generelle Aussage über die zu erwartende Leistung von Continual Pretraining machen. Das Training der Modelle ist durch die Ressourcenanforderungen begrenzt und stellt auch nicht die optimale Leistung dar, die ein Llama 2 Modell erreichen kann.

Während des Trainings wurde ein Zustand der Überanpassung erreicht, wobei der genaue Zeitpunkt nicht bestimmt werden konnte. Bereits nach 3 Epochen ist ein deutlicher Anstieg der Fehlerwerte gegenüber dem Validierungsdatensatz zu erkennen, so dass davon auszugehen ist, dass dies bereits vorher begonnen hat. Zudem wurde die Unterteilung des Fortschritts nur in 1, 3, 5 und 10 Epochen vorgenommen, weshalb nicht ausgeschlossen werden kann, dass z.B. bei 4 Epochen bessere Ergebnisse erzielt werden können.

Auch die geringe Datenmenge für das Training reduziert die mögliche Leistung in allen Kriterien. Kleine Datenmengen führen zu Überanpassung und stellen Informationen in zu wenig unterschiedlichen Ausprägungen dar, was die Generalisierbarkeit des Modells verringert.

Die Evaluierung des Modells erfolgte anhand von 95 Fragen. Auch dieser Fragenkatalog ist kleiner als erwartet und enthält meist ähnliche Wissensfragen in nur einer Formulierung. Insbesondere die Fragen aus der schriftlichen Prüfung des Moduls „Informationssysteme im Gesundheitswesen“ umfassen insgesamt nur 9 Fragen und unterliegen damit sehr starken Schwankungen.

Die Ergebnisse der Evaluation zeigen eine Tendenz zur Verbesserung, die trainierten Modelle selbst sind in diesem Zustand jedoch nicht verwendbar. Dies liegt zum einen an dem fehlenden Finetuning, welches die Gefährlichkeit des Modells deutlich erhöht, da es schädliche und falsche Antworten generieren kann, zum anderen an der allgemeinen Formulierung der Antworten. Wie bereits in Kapitel 6 gezeigt, beenden Modelle ihre Antwort nicht eindeutig mit einem Token, sondern generieren fortlaufend Text. Daher enthalten alle generierten Antworten zumeist am Ende zitierte Textpassagen, die keinen Bezug zur Frage haben. Dies erschwert die Verwendung dieses Modells erheblich. Abhilfe schafft hier das bereits erwähnte Finetuning, bei dem einem Modell abschließende Token beigebracht werden, nach denen die Generierung gestoppt werden kann.

7.2 PROBLEME BEI KERNFRAGEN

Das Thema Informationssysteme im Gesundheitswesen und damit auch der hier verwendete Evaluationsdatensatz enthalten grundlegende Kerninformationen, die in ihrem Verständnis wichtiger sind als anderes Wissen. Dazu gehören z.B. die Definition von Daten, Information und Wissen, die Definition eines Informationssystems, so dass Menschen Teil des Systems sind, oder die Unterscheidung zwischen Anwendungssystemen und Softwareprodukten (ein Anwendungssystem ist die Installation und Nutzung eines Softwareprodukts durch Akteure, deren Rollen und Aufgaben).

Diese grundlegenden Wissensfragen werden in der Literatur zu diesem Thema unterschiedlich definiert und sollte von den Modellen mit dem Wissen aus dem Buch von Winter u. a. (2023) beantwortet werden. Auch widersprechen einige Definitionen aus dem Buch der Definition aus dem allgemeinen Sprachgebrauch. Auch hier ist eine Beantwortung mit dem Wissen aus dem Buch erforderlich.

Die hier trainierten Modelle wurden jedoch auf Literatur vortrainiert, die durchaus widersprüchliches Wissen zum Buch enthält. So beantworten nicht weitertrainierte Modelle Kernfragen wie „Was ist ein Informationssystem“ oft falsch und beziehen den Menschen nicht mit ein. Mit fortschreitender Epoche verbessert sich dieses Verhalten, da die Modelle den Zustand der Überanpassung erreichen und nur noch Wissen aus dem Buch zitieren. Dennoch kann eine grundsätzliche Falschbeantwortung von Kernfragen in dieser Arbeit nicht ausgeschlossen werden. Hier wäre eine genauere Evaluierung mit nur Kernfragen notwendig, um die Leistung der Modelle in diesem Bereich zu untersuchen.

7.3 BEWERTUNG DER FRAGEN MIT PRÜFUNGSPUNKTEN

Zur Berechnung des Kriteriums Korrektheit wurden jeder Frage drei Werte zugeordnet: die Anzahl der korrekten Antworten des Modells, die Anzahl der Antworten des Modells im Allgemeinen und die Anzahl der erwarteten Antworten für diese Frage. Anhand dieser Werte wurden die F1- und MakroF1-Werte der Modelle berechnet.

Da der Evaluationsdatensatz jedoch größtenteils aus Prüfungsfragen besteht, wäre hier auch eine Auswertung der Fragen nach den Original-Prüfungspunkten möglich. Dadurch würden schwierigere Fragen bzw. Fragen mit mehr Antworten stärker gewichtet und hätten somit einen größeren Einfluss auf die Leistung der Modelle. Diese Art der Auswertung könnte die Modellleistung in den Kontext der menschlichen Leistung stellen und die Aussagen über die Modellleistung nachvollziehbarer machen.

7.4 LÖSUNG DES PROBLEMS

Das in Abschnitt 1.2 formulierte Problem kann mit den hier vorgestellten Modellen nur teilweise gelöst werden. Die Verwendung des Modells erleichtert die Wissensextraktion aus dem Buch, gibt aber in weiten Teilen den Inhalt des Buches nicht oder nur unvollständig wieder. Mit zunehmender Epoche der Modelle steigt die Qualität der Wissensextraktion durch Zitieren des Buches, jedoch sinkt die Anwendbarkeit auf andere Formulierungen und die Verallgemeinerung des gleichen Wissens, insbesondere bei der Verwendung in einem anderen Kontext. Die Fragmentierung der Definitionen wird durch das Modell zusammengeführt, was sich insbesondere an den Ergebnissen der Auswertung bei den Fragen zum Buch zeigt. Hier gelang es den Modellen, die im Buch enthaltenen Wissensfragen mit zunehmender Epoche den entsprechenden Antworten zuzuordnen.

Das Ziel Z1 wurde mit den hier vorgestellten Modellen teilweise erreicht. Die trainierten Modelle sind in der Lage, Fragen zu Informationssystemen im Gesundheitswesen mit einem MakroF1-Wert von 0,3 korrekt zu beantworten, steigern damit die Ausgangsergebnisse des untrainierten Modells, fallen aber in ihrer Leistung hinter das SOTA Modell GPT4 zurück.

Ziel Z2 wird wie Ziel Z1 teilweise erreicht. Das Modell Llama 2 7B, trainiert mit Nvidia Tesla A30 Grafikkarten über 3 Epochen, erreicht bei den Klausurfragen des Moduls „Architektur von Informationssystemen im Gesundheitswesen“ einen MakroF1-Wert von 0,45 und beantwortet 12 von 22 Fragen mit mindestens einer richtigen Antwort. Diese Ergebnisse liegen weit über dem untrainierten Modell mit einem

MakroF1-Wert von 0,12 und 6 von 22 richtig beantworteten Fragen, werden aber immer noch vom GPT4-Modell mit einem MakroF1-Wert von 0,64 und 17 von 22 richtig beantworteten Fragen geschlagen. Die hier vorgestellten Ergebnisse zeigen, dass die Leistung bei der Lösung dieser Beispielklausur durch Continual Pretraining deutlich gesteigert werden kann, jedoch durch die Größe des Modells und den verwendeten Datensatz begrenzt ist.

8 AUSBLICK

Möglichkeiten zur Verbesserung der Leistung von Sprachmodellen auf die hier vorgestellten Ziele bieten sich in vielerlei Hinsicht. Eine Auswahl der erfolgsversprechensten Ansätze wird in diesem Kapitel vorgestellt.

8.1 MODELLVERGRÖSSERUNG

Die intuitivste Lösung ist die Vergrößerung der verwendeten Modelle. Von dem verwendeten Llama 2 Modell gibt es auch Modelle mit 13 Milliarden und 70 Milliarden Parametern. Bereits im Artikel Touvron u. a. (2023a) über Llama 1 Modelle erreicht nur das größte Modell eine vergleichbare Leistung wie ChatGPT. Es ist also davon auszugehen, dass auch hier größere Modelle eine bessere Leistung erzielen können. Kaplan u. a. (2020) haben gezeigt, dass die Leistung eines Modells neben der Größe des Datensatzes und der Länge des Trainings auch von der Größe des Modells abhängt. Dies unterstützt die hier vorgestellte These.

Allerdings bringt eine Vergrößerung des Modells auch Nachteile und Probleme mit sich. Größere Modelle benötigen mehr Rechenleistung und GPU-RAM während des Trainings und der Inferenz. Dies führt zu höheren Kosten und längeren Trainingszeiten und schränkt die Verwendbarkeit der Modelle nach dem Training deutlich ein. In der hier verwendeten Konfiguration konnte die Anzahl der verwendeten GPUs nicht erhöht werden, daher war ein Training größerer Modelle nicht möglich.

Größere Modelle neigen schneller dazu, einen Zustand der Überanpassung zu erreichen, weshalb das Training mit einer deutlich geringeren Lernrate durchgeführt wird, was die Trainingszeit weiter erhöht.

8.1.1 *Training durch Quantisierung*

Um große Modelle zu trainieren und zu nutzen, ist der Einsatz der ZeRO-Optimierung ein erster Schritt, der in Rajbhandari u. a. (2020) vorgestellt wird. Mit Hilfe der ZeRO-Stufe 3 Optimierung können Optimierungs- und Parameterzustände auf CPUs und Festplatten ausgelagert werden, während die Berechnungen weiterhin auf GPUs durchgeführt werden. Zusätzlich kann durch eine Quantisierung der Mo-

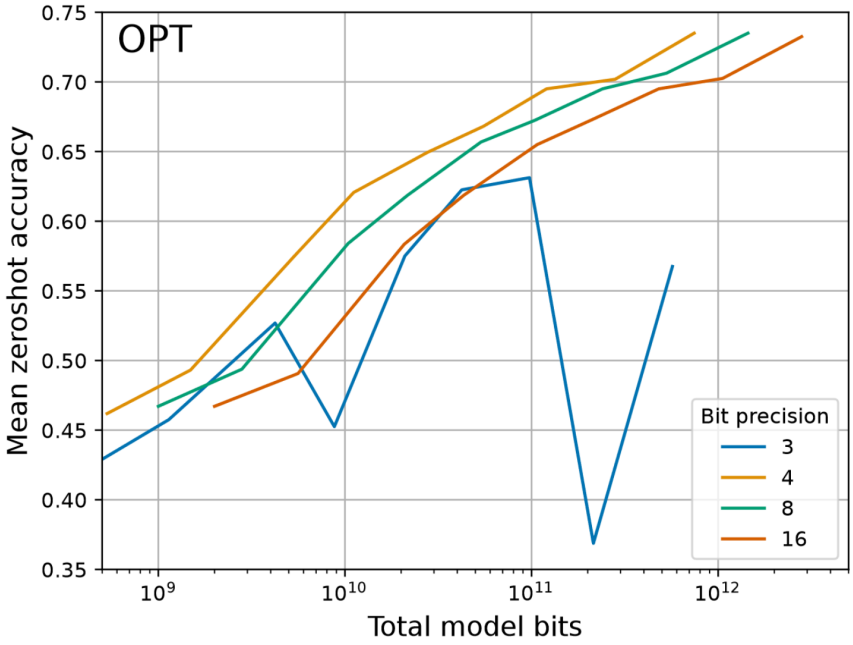

Abbildung 8.1: Korrektheit von OPT Modellen aufgeteilt nach unterschiedlicher Bitanzahl pro Parameter aus Dettmers und Zettlemoyer (2023). Modelle mit gleicher Parameteranzahl liegen ungefähr auf gleicher X-Achse, jedoch nicht auf gleicher Y-Achse.

delle in kleinere Datentypen die Anzahl der benötigten GPU-RAMs deutlich reduziert werden. Dettmers und Zettlemoyer (2023) untersuchte den Einfluss der Quantisierung auf die Performance der Modelle und zeigte, dass eine Quantisierung auf 4 Bit die Leistung der Modelle nur geringfügig beeinflusst. Abb. 8.1 zeigt die Korrektheit von OPT-Modellen mit unterschiedlichen Datentypen (3, 4, 8 und 16 Bit). Modelle mit gleicher Parameteranzahl liegen nicht auf dem gleichen Y-Achse, da die X-Achse in Modellbits angegeben ist. Modelle mit der gleichen Anzahl von Parametern, aber einer geringeren Anzahl von Bits pro Parameter sind daher weiter links in der Grafik angeordnet. Dennoch zeigen diese Abbildung von Dettmers und Zettlemoyer (2023) und ähnliche Abbildungen für andere Modelltypen eine fast identische Leistung bis einschließlich 4 Bit.

Zusätzlich ist es möglich, Modelle während der Inferenz in einen 8-Bit-Modus zu konvertieren, um Ressourcen zu sparen. Dettmers u. a. (2022) haben gezeigt, dass bei der Verwendung von 8-Bit-Matrix-Multiplikationen keine signifikanten Leistungseinbußen zu erwarten sind. Der 8-Bit Inferenzmodus wurde auch in dieser Arbeit verwendet, um die trainierten Modelle auf einer Nvidia Tesla A30 GPU mit 24 GB RAM nutzen zu können.

8.2 HUMAN REINFORCEMENT LEARNING

Nach erfolgreichem Training eines Modells kann dessen Ausgabe wesentlich verbessert werden, indem Menschen die Ausgabe bewerten und das Modell entsprechend anpassen. Dieser Schritt wurde auch bei GPT4 durchgeführt und ist einer der Gründe für die hohe Leistungsfähigkeit des Modells. Mit Hilfe des so genannten Human Reinforcement Learning können Modelle darauf trainiert werden, Erklärungen zu ihren Antworten zu geben, unbeantwortete Fragen mit z.B. der generierten Antwort „Ich weiß es nicht“ zu markieren, auf schädliche Inhalte nicht zu reagieren oder die Antwort zu verweigern und Texte klarer und freundlicher zu formulieren. Dieser Trainingsschritt ist jedoch sehr aufwendig und kann nicht vollständig automatisiert werden.

Llama 2 verfügt über Versionen, die dieses Human Reinforcement Learning bereits durchlaufen haben. In dieser Arbeit wird davon ausgegangen, dass dieses zusätzlich erlernte Wissen verloren geht, sobald ein Continual Pretraining durchgeführt wird. Diese Annahme basiert auf der Tatsache, dass populäre Modelle das Human Reinforcement Learning als letzten Trainingsschritt durchlaufen haben. Eine Widerlegung dieser Hypothese könnte potenziell auch die Leistung der Modelle verbessern.

8.3 DATENSATZVERGRÖSSERUNG

Eine wesentliche Einschränkung der hier trainierten Modelle ist die Größe des Trainingsdatensatzes. Mit etwa 34 500 Tokens ist dieser Datensatz nicht mit den für das ursprüngliche Training verwendeten Datensätzen vergleichbar und erreicht nur eine Größe von 600 kB. Kleine Datensätze führen zu einer schnelleren Überanpassung, da der Datensatz mehrfach verwendet werden muss, bevor eine Anpassung des Modells an diesen erfolgen kann. Dies zeigt sich auch in den Ergebnissen, da bereits nach 3 Epochen eine Überanpassung zu erkennen ist. Größere Datensätze führen zu weniger Epochen und damit zu weniger Überanpassung. Zudem enthalten größere Datensätze potentiell gleiche Informationen in unterschiedlicher Formulierung, was wiederum die Generalisierbarkeit des Modells verbessert. Insbesondere die Kriterien Robustheit und Fragenverständnis profitieren von größeren Datensätzen.

Für die Verwendung größerer Datensätze wäre es optimal, weitere Literatur zum Thema Informationssysteme im Gesundheitswesen heranzuziehen. Wichtig ist dabei, dass die Wissensinhalte identisch sind, so dass sich die Aussagen zwischen den Büchern nicht widersprechen. Auch eine Zusammenstellung einzelner Kapitel aus verschiedenen

Quellen mit gleichem Inhalt könnte die Größe des Datensatzes erhöhen und damit die Leistung der Modelle verbessern.

8.4 DOMÄNENSPEZIFISCHE MODELLE

Das Llama 2 Modell wurde ohne speziellen Fokus auf eine Domäne, insbesondere die Domäne Medizininformatik, trainiert. Allgemein trainierte Modelle können einen breiteren Anwendungsbereich abdecken, enthalten aber auch weniger Verständnis einzelner Domänen als domänenspezifische Modelle. Die Verwendung von Modellen, die stärker auf die Domäne der Medizininformatik oder auf wissenschaftliche Inhalte fokussiert sind, könnte die Basisleistung des Modells und damit auch die Endleistung der hier trainierten Modelle verbessern.

Leider existieren zum Zeitpunkt dieser Arbeit keine domänenspezifischen Modelle in der Größe von Llama 2, da entweder die Größe der Modelle oder die Größe des Datensatzes geringer ist. Domänenspezifische Modelle können auch Informationen enthalten, die im Widerspruch zum verwendeten Trainingsdatensatz stehen, aber besser gelernt wurden, weil mehr Informationen im ursprünglichen Datensatz vorhanden waren. Das bewusste Verlernen dieses falschen Wissens ist nicht trivial und kann die Leistung des Modells negativ beeinflussen. Eine Möglichkeit der Modellanpassung wurde von Dai u. a. (2022) vorgestellt, um bestimmte Wissensneuronen aus einem Modell zu entfernen.

8.5 ADAPTER-BASIERTES TRAINING

Wie bereits in Kapitel 3 erwähnt, ist neben den klassischen Trainingsmethoden auch das Training mit Hilfe von Adaptern möglich. Pfeiffer u. a. (2020) stellt eine Plattform zur Verfügung, die es erlaubt, kleinere Neuronale Netze zwischen die Modellschichten einzufügen und diese zu trainieren. Dies minimiert die benötigten Ressourcen, verkürzt die Trainingszeit und verhindert das Verlernen von Wissen. Adapter müssen speziell für Modelltypen entwickelt werden und sind zum Zeitpunkt dieser Arbeit nicht für Llama-Modelle verfügbar. Eine eigene Implementierung für die Llama Modelle könnte die endgültige Performance der Modelle verbessern.

Neben dem Einfügen von Adaptern ist auch die Verwendung von Low-Rank Adapation (LoRA), eingeführt in Hu u. a. (2022), möglich. LoRA erlaubt das Einfügen von trainierbaren Matrizen in jede Schicht und hat ähnliche Vorteile wie Adapter. Die Verwendung von LoRA wurde im Rahmen dieser Arbeit untersucht, jedoch nicht in die endgültige Auswertung übernommen, da eine Inferenz nicht möglich war.

8.6 TEXTEXTRAKTION AUS KONTEXT

Diese Arbeit konzentriert sich primär auf die Verwendung von Decoder-basierten Modellen zur Generierung von Antworten auf Fragen. Grund dafür ist die potentiell höhere Generalisierbarkeit auf Formulierungen um besser generierten Text zu erhalten. Neben der Generierung von Text ist auch die Extraktion von Text aus einem gegebenen Kontext eine mögliche Lösung. Mit Hilfe von Encoder-basierten Modellen wie z.B. BERT ist es möglich, Textpassagen, die Antworten auf gegebene Fragen enthalten, aus einem Kontext zu extrahieren. Dieser Ansatz könnte zu einer besseren Korrektheit der Fragen führen, allerdings könnten andere Kriterien wie Robustheit und Erklärbarkeit darunter leiden.

8.7 RETRIEVAL AUGMENTED GENERATION

Man könnte auch ganz auf das Training von Decoder-basierten Modellen verzichten, bei denen das Modell die Frage nur mit Hilfe eines gegebenen Kontextes beantworten soll. Dieser Ansatz verspricht sehr gute Ergebnisse, da Modelle wie GPT4 bereits sehr gute Leistungen bei der Extraktion von Informationen aus Kontexten zeigen.

Liu u. a. (2023) haben dieses Verhalten an großen Sprachmodellen untersucht und gezeigt, dass Modelle in der Lage sind, Informationen aus Anfang und Ende eines Kontextes weitgehend korrekt zu extrahieren. Um nur den Kontext zur Beantwortung von Fragen zu verwenden, müssen Modelle mit einer großen Kontextlänge gewählt und das verwendete Buch optional gekürzt werden. Chen u. a. (2023) beschreibt eine Möglichkeit, die verfügbare Kontextlänge mit Hilfe von positionsabhängiger Interpolation zu erweitern.
Eine Kombination aus einem Encoder-basierten Modell zur Bestimmung der Textposition und einem Decoder-basierten Modell zur Generierung einer Antwort mit Hilfe der extrahierten Textposition könnte ebenfalls Abhilfe schaffen. Dieser Ansatz wurde von Lewis u. a. (2020) untersucht und erzielte SOTA Ergebnisse in drei QA Evaluationsdatensätzen. Ähnliche Ergebnisse wurden von Mao u. a. (2020) in ihrem Artikel erreicht.

Dieser Ansatz erfordert kein weiteres Training von Modellen und führt dementsprechend zu deutlich weniger Leistungsanforderungen. Das bedeutet, dass größere Modelle unter gleicher Hardware genutzt werden können und damit auch die erwarteten Ergebnisse höher ausfallen könnten. Die Komplexität liegt hierbei in der Integration der Modelle zu einem Gesamtsystem und der Veränderung bereits existierender Modelle um größere Kontextlängen zu ermöglichen.

Ansatz	Aufwand	Erfolgseinschätzung
Retrieval Augmented Generation	hoch	sehr hoch
Datensatzvergrößerung	niedrig	hoch
Modellvergrößerung	hoch	hoch
Human Reinforcement Learning	sehr hoch	hoch
Domänenspezifische Modelle	niedrig	mittel
Textextraktion aus Kontext	niedrig	mittel
Adapter-basiertes Training	mittel	mittel

Tabelle 8.1: Übersicht über mögliche Ansätze zur Verbesserung der Leistung der Modelle

8.8 ZUSAMMENFASSUNG

Die hier vorgestellten Richtungen für zukünftige Arbeiten versprechen eine Verbesserung der Leistung der Modelle in vielerlei Hinsicht. Zum besseren Überblick stell Tabelle 8.1 eine Übersicht über die hier vorgestellten Ansätze dar. Die Tabelle ist nach Erfolgseinschätzung sortiert. Ein hoher Aufwand bedeutet grundlegende strukturelle Änderungen im genutzten Prozess und erfordert dementsprechend hohen Zeit- und Kostenaufwand.

ZUSAMMENFASSUNG

Die Wissensbeschaffung in der Medizininformatik ist ein komplexer und wichtiger Bestandteil von Forschung, Lehre und Praxis. Umso wichtiger ist die Effizienz und Qualität der Wissensbeschaffung. Informationssysteme stellen dabei eine grundlegende Architektur in diesem Bereich dar. Das Buch *Health Information Systems* von Winter u. a. (2023) enthält eine umfassende Einführung in die Thematik der Informationssysteme in der Medizin. Die Extraktion von Informationen und Wissen aus dem Buch erweist sich jedoch als schwierig. In dieser Arbeit wurde ein vortrainiertes Sprachmodell verwendet, um den Inhalt des Buches effizienter und leichter zugänglich zu machen. Ziel ist es, Wissen aus dem Buch zu extrahieren. Dieses Ziel wurde anhand von Prüfungsfragen aus dem Buch und Modulen der Universität Leipzig, die inhaltlich auf dem Buch aufbauen, evaluiert.

Um diese Ziele zu erreichen, wurden verschiedene Sprachmodelle auf ihre Eignung verglichen. Die Wahl fiel auf das Sprachmodell Llama 2 von MetaAI (Touvron u. a., 2023b). Dieses vortrainierte Sprachmodell wurde mit Hilfe des Continual Pretrainings weiter auf das Buch trainiert. Während des Trainings wurde die Qualität des Modells zu verschiedenen Zeitpunkten evaluiert. Diese Bewertung basierte auf den Kriterien Korrektheit, Erklärbarkeit, Fragenverständnis und Robustheit. Abschließend wurde ein Vergleich zwischen den Trainingszeitpunkten, dem nicht weiter trainierten Modell und dem SOTA Modell GPT4 durchgeführt.

Die Ergebnisse zeigen, dass Modelle mit Hilfe von Continual Pretraining auf ein spezifisches Buch trainiert werden können und in fast allen Kriterien bessere Ergebnisse erzielen als das nicht weiter trainierte Modell. Aufgrund des Größenunterschieds zwischen GPT4 und dem hier verwendeten Modell Llama 2 7B wird die Leistung von GPT4 jedoch nicht erreicht. Das nicht weiter trainierte Modell erreichte nur einen MakroF1-Wert von 0,1 und konnte durch das hier durchgeführte Training auf 0,33 gesteigert werden. Dies zeigt das Potential dieser Methode. Allerdings sind die Modelle noch nicht mit dem SOTA Modell GPT4 vergleichbar, das einen MakroF1-Wert von 0,7 erreichte. Die hier trainierten Modelle stellen keinen für die Praxis nutzbaren Zustand dar, lösen aber die gestellte Zielstellung „Machbarkeit der Beantwortung von Fragen mit Hilfe von Sprachmodellen“. Ausführliche Ergebnisgrafiken und die verwendeten Skripte stehen unter `https://doi.org/10.5281/zenodo.8363501` zur Verfügung.

Um die Leistungsfähigkeit von Sprachmodellen zu erhöhen, können sowohl die Modellgröße als auch die Größe des Datensatzes erhöht werden. Auch neue Entwicklungen im Bereich der Adapter (Abschnitt 8.5) zeigen ressourceneffiziente Ansätze, um die Performanz von Sprachmodellen zu steigern. Insbesondere der Einsatz von Retrieval-Augmented Generation bieten vielversprechende Erweiterungsmöglichkeiten.

Zusammenfassend kann gesagt werden, dass die Leistung von Sprachmodellen bei der Beantwortung von Klausurfragen durch Continual Pretraining gesteigert werden kann, auch wenn die Endleistung noch keinen praxistauglichen Zustand erreicht. So ist die Lösung einer Beispielklausur durch die Modelle nicht vollständig möglich, ebenso können Fragen zu Informationssystemen im Gesundheitswesen nur teilweise beantwortet werden. Die Schwierigkeit der Wissensbeschaffung aus Winter u. a. (2023) kann durch die Modelle erleichtert werden, insbesondere die Zusammenführung fragmentierter Definitionen, erreicht aber keinen in der Praxis nutzbaren Zustand, weshalb die Wissensbeschaffung aus dem Buch weiterhin schwierig bleibt.

LITERATUR

Ahrens, Wolfgang u. a. (2023). *Ethische Leitlininen der Deutschen Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie e.V. (GMDS), des Arbeitskreises der IT-Leiter/innen der Universitätsklinika (AL-KRZ) des Berufsverbandes Medizinischer Informatiker (BVMI), des Bundesverbandes der Krankenhaus-IT-Leiterinnen/Leiter e.V. (KH-IT) und des Deutschen Verbandes Medizinischer Dokumentare e.V. (DVMD)*. URL: https://www.gmds.de/fileadmin/user_upload/Aktivitaeten_Themen/praesidiumskommissionen/Ethische_Leitlinien.pdf (besucht am 23. 04. 2023).

Black, Sidney u. a. (Mai 2022). „GPT-NeoX-20B: An Open-Source Autoregressive Language Model“. In: *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*. Association for Computational Linguistics, S. 95–136. DOI: 10.18653/v1/2022.bigscience-1.9.

Brasil, Sandra, Carlota Pascoal, Rita Francisco, Vanessa dos Reis Ferreira, Paula A. Videira und Gonçalo Valadão (2019). „Artificial Intelligence (AI) in Rare Diseases: Is the Future Brighter?“ In: *Genes* 10.12. ISSN: 2073-4425. DOI: 10.3390/genes10120978.

Brown, Tom u. a. (2020). „Language Models are Few-Shot Learners“. In: *Advances in Neural Information Processing Systems*. Hrsg. von H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan und H. Lin. Bd. 33. Curran Associates, Inc., S. 1877–1901.

Brunsch, Hannes R. (2018). „Question Answering auf SNIK“. In: *SKILL 2018-Studentenkonferenz Informatik*.

Brunsch, Hannes Raphael (2022). „Question Answering auf SNIK“. Besondere Lernleistung. Leipzig, Germany: Wilhelm-Ostwald-Schule. URL: https://www.snik.eu/public/bell-hrb.pdf.

Chen, Shouyuan, Sherman Wong, Liangjian Chen und Yuandong Tian (2023). *Extending Context Window of Large Language Models via Positional Interpolation*. arXiv: 2306.15595 [cs.CL].

Dai, Damai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang und Furu Wei (2022). *Knowledge Neurons in Pretrained Transformers*. arXiv: 2104.08696 [cs.CL].

Davenport, Thomas, Abhijit Guha, Dhruv Grewal und Timna Bressgott (2020). „How artificial intelligence will change the future of marketing“. In: *Journal of the Academy of Marketing Science* 48.1, S. 24–42. ISSN: 1552-7824. DOI: 10.1007/s11747-019-00696-0.

Dehouche, Nassim (März 2021). „Plagiarism in the age of massive Generative Pre-trained Transformers (GPT-3): “The best time to act was yesterday. The next best time is now.”“ In: *Ethics in Science and Environmental Politics* 21. DOI: 10.3354/esep00195.

Dettmers, Tim, Mike Lewis, Younes Belkada und Luke Zettlemoyer (2022). „LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale“. In: *CoRR* abs/2208.07339. URL: https://doi.org/10.48550/arXiv.2208.07339.

Dettmers, Tim und Luke Zettlemoyer (2023). „The case for 4-bit precision: k-bit Inference Scaling Laws“. In: *Proceedings of the 40th International Conference on Machine Learning*. Hrsg. von Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato und Jonathan Scarlett. Bd. 202. Proceedings of Machine Learning Research. PMLR, S. 7750–7774. URL: https://proceedings.mlr.press/v202/dettmers23a.html.

Esteva, Andre, Brett Kuprel, Roberto A. Novoa u. a. (2017). „Dermatologist-level classification of skin cancer with deep neural networks“. In: *Nature* 542, S. 115–118. DOI: 10.1038/nature21056.

Geng, Xinyang und Hao Liu (Mai 2023). *OpenLLaMA: An Open Reproduction of LLaMA*. URL: https://github.com/openlm-research/open_llama.

Gururangan, Suchin, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey und Noah A. Smith (Juli 2020). „Don't Stop Pretraining: Adapt Language Models to Domains and Tasks“. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, S. 8342–8360. DOI: 10.18653/v1/2020.acl-main.740. URL: https://aclanthology.org/2020.acl-main.740.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren und Jian Sun (2016). „Deep Residual Learning for Image Recognition“. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, S. 770–778. DOI: 10.1109/CVPR.2016.90.

Houlsby, Neil, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan und Sylvain Gelly (2019). „Parameter-Efficient Transfer Learning for NLP“. In: *Proceedings of the 36th International Conference on Machine Learning*. Hrsg. von Kamalika Chaudhuri und Ruslan Salakhutdinov. Bd. 97. Proceedings of Machine Learning Research. PMLR, S. 2790–2799. URL: https://proceedings.mlr.press/v97/houlsby19a.html.

Hu, Edward J, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang und Weizhu Chen (2022). „LoRA: Low-Rank Adaptation of Large Language Models“. In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=nZeVKeeFYf9.

Jahn, Franziska, Michael Schaaf, Barbara Paech und Alfred Winter (2014). „Ein Semantisches Netz des Informationsmanagements im Krankenhaus“. In: *Informatik 2014*. Hrsg. von E. Plödereder, L. Grunske, E. Schneider und D. Ull. Lecture Notes in Informatics. Bonn: Gesellschaft für Informatik e.V., S. 1491–1498.

Jiang, Zhengbao, Antonios Anastasopoulos, Jun Araki, Haibo Ding und Graham Neubig (Nov. 2020). „X-FACTR: Multilingual Factual Knowledge Retrieval from Pretrained Language Models“. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, S. 5943–5959. DOI: 10.18653/v1/2020.emnlp-main.479.

Johnson, Kevin B., Wei-Qi Wei, Dilhan Weeraratne, Mark E. Frisse, Karl Misulis, Kyu Rhee, Juan Zhao und Jane L. Snowdon (2021). „Precision Medicine, AI, and the Future of Personalized Health Care“. In: *Clinical and Translational Science* 14.1, S. 86–93. DOI: 10.1111/cts.12884.

Kalyan, Katikapalli Subramanyam, Ajit Rajasekharan und Sivanesan Sangeetha (2022). „AMMU: A survey of transformer-based biomedical pretrained language models“. In: *Journal of biomedical informatics* 126, S. 103982. DOI: 10.1016/j.jbi.2021.103982.

Kaplan, Jared, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu und Dario Amodei (2020). *Scaling Laws for Neural Language Models*. arXiv: 2001.08361 [cs.LG].

Kinnebrock, Werner (1994). *Neuronale Netze: Grundlagen, Anwendungen, Beispiele*. German. 2., verbesserte Auflage. Reprint 2018. Berlin: Oldenbourg Wissenschaftsverlag. ISBN: 9783486786361. URL: https://doi.org/10.1515/9783486786361.

Kudo, Taku und John Richardson (Nov. 2018). „SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing“. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Brussels, Belgium: Association for Computational Linguistics, S. 66–71. DOI: 10.18653/v1/D18-2012. URL: https://aclanthology.org/D18-2012.

Lee, Jinhyuk, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So und Jaewoo Kang (2019). „BioBERT: a pre-trained biomedical language representation model for biomedical text mining“. In: *Bioinformatics* 36.4, S. 1234–1240. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz682.

Lewis, Patrick u. a. (2020). „Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks“. In: *ArXiv* abs/2005.11401. URL: https://api.semanticscholar.org/CorpusID:218869575.

Lhoest, Quentin u. a. (Nov. 2021). „Datasets: A Community Library for Natural Language Processing“. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online und Punta Cana, Dominican Republic: Association for Computational Linguistics, S. 175–184. arXiv: 2109.02846 [cs.CL]. URL: https://aclanthology.org/2021.emnlp-demo.21.

Liu, Nelson F., Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni und Percy Liang (2023). *Lost in the Middle: How Language Models Use Long Contexts*. arXiv: 2307.03172 [cs.CL].

Loshchilov, Ilya und Frank Hutter (2019). „Decoupled Weight Decay Regularization". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

Mao, Yuning, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han und Weizhu Chen (2020). „Generation-Augmented Retrieval for Open-Domain Question Answering". In: *Annual Meeting of the Association for Computational Linguistics*. URL: https://api.semanticscholar.org/CorpusID:221802772.

McCulloch, Warren und Walter Pitts (1943). „A logical calculus of the ideas immanent in nervous activity". In: Bd. 5. 4. The bulletin of mathematical biophysics, S. 115–133. DOI: 10.1007/BF02478259.

Micikevicius, Paulius u. a. (2018). „Mixed Precision Training". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=r1gs9JgRZ.

Omar, Reham, Omij Mangukiya, Panos Kalnis und Essam Mansour (2023). *ChatGPT versus Traditional Question Answering for Knowledge Graphs: Current Status and Future Directions Towards Knowledge Graph Chatbots*. Version 1. arXiv: 2302.06466 [cs.CL].

OpenAI (2023). „GPT-4 Technical Report". In: arXiv: 2303.08774 [cs.CL].

Paszke, Adam u. a. (2019). „PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc.

Petroni, Fabio, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu und Alexander Miller (Nov. 2019). „Language Models as Knowledge Bases?" In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, S. 2463–2473. DOI: 10.18653/v1/D19-1250.

Pfeiffer, Jonas, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho und Iryna Gurevych (Okt. 2020). „AdapterHub: A Framework for Adapting Transformers". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, S. 46–54. DOI: 10.18653/v1/2020.emnlp-demos.7. URL: https://aclanthology.org/2020.emnlp-demos.7.

QAnswer (2023). *Question Answering*. URL: https://app.qanswer.ai/public-share?kb=SNIK_BB&type=graph&user=kirdie (besucht am 09. 03. 2023).

Radford, Alec und Karthik Narasimhan (2018). „Improving Language Understanding by Generative Pre-Training“. In.

Radford, Alec, Jeff Wu, Rewon Child, David Luan, Dario Amodei und Ilya Sutskever (2019). „Language Models are Unsupervised Multitask Learners“. In: URL: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

Rajbhandari, Samyam, Jeff Rasley, Olatunji Ruwase und Yuxiong He (2020). „ZeRO: Memory Optimizations toward Training Trillion Parameter Models“. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '20. Atlanta, Georgia: IEEE Press. ISBN: 9781728199986.

Rumelhart, David E., Geoffrey E. Hinton und Ronald J. Williams (1986). „Learning representations by back-propagating errors“. In: Bd. 323. 6088. Nature, S. 533–536. DOI: 10.1038/323533a0.

Schaaf, Michael, Franziska Jahn, Kais Tahar, Christian Kucherer, Alfred Winter und Barbara Paech (2015). „Entwicklung und Einsatz einer Domänenontologie des Informationsmanagements im Krankenhaus“. In: *Informatik 2015*. Lecture Notes in Informatics 246. Hrsg. von Douglas W. Cunningham, Petra Hofstedt, Klaus Meer und Ingo Schmitt.

Sennrich, Rico, Barry Haddow und Alexandra Birch (2016). „Neural Machine Translation of Rare Words with Subword Units“. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, S. 1715–1725. DOI: 10.18653/v1/P16-1162. URL: https://aclanthology.org/P16-1162.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever und Ruslan Salakhutdinov (2014). „Dropout: A Simple Way to Prevent Neural Networks from Overfitting“. In: *Journal of Machine Learning Research* 15.56, S. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

Touvron, Hugo u. a. (2023a). *LLaMA: Open and Efficient Foundation Language Models*. arXiv: 2302.13971 [cs.CL].

Touvron, Hugo u. a. (2023b). *Llama 2: Open Foundation and Fine-Tuned Chat Models*. arXiv: 2307.09288 [cs.CL].

Usbeck, Ricardo, Michael Röder, Michael Hoffmann, Felix Conrads, Jonathan Huthmann, Axel-Cyrille Ngonga-Ngomo, Christian Demmler und Christina Unger (2019). „Benchmarking question answering systems“. In: *Semantic Web* 10.2, S. 293–304.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser und Illia Polosukhin (2017). „Attention is All you Need“. In: *Advances in Neural Information Processing Systems*. Hrsg. von I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan und R. Garnett. Bd. 30. Curran Associates, Inc.

Wang, Ben und Aran Komatsuzaki (Mai 2021). *GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model*. https://github.com/kingoflolz/mesh-transformer-jax.

Winter, Alfred, Elske Ammenwerth, Reinhold Haux, Michael Marschollek, Bianca Steiner und Franziska Jahn (2023). *Health Information Systems*. 3. Aufl. Health Informatics. Springer Cham. ISBN: 978-3-031-12310-8. DOI: 10.1007/978-3-031-12310-8. URL: https://link.springer.com/book/10.1007/978-3-031-12310-8.

Wolf, Thomas u. a. (Okt. 2020). „Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, S. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Ziegler, Daniel M., Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul F. Christiano und Geoffrey Irving (2019). „Fine-Tuning Language Models from Human Preferences". In: *CoRR* abs/1909.08593. URL: http://arxiv.org/abs/1909.08593.

APPENDIX

APPENDIX

EVALUIERUNGSDATENSATZ

Die für das Training und die Evaluierung der Modelle verwendeten Skripte sind unter `https://doi.org/10.5281/zenodo.8363501` verfügbar. Einige Datensätze der Fragen sind nicht enthalten, da sie nicht öffentlich zugänglich sind. Dieser eingeschränkte Teil ist mit den erforderlichen Zugriffsrechten unter `https://github.com/scorixear/master-thesis-database` zu finden.

ERKLÄRUNG

Ich versichere, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann. Ich versichere, dass das elektronische Exemplar mit den gedruckten Exemplaren übereinstimmt.

Leipzig, 30.09.2023

Paul Keller