

DEversAI: Training und Visualisierung deutsch lokalizierter direktionalkomplementärer LLMs

Leo Blume (16 J.)

Projektbetreuer: Prof. Dr. André Grüning

Erarbeitungsort: Hochschule Stralsund (HOST)

Fachgebiet: Mathematik/Informatik

Bundesland: Mecklenburg-Vorpommern

Wettbewerbsjahr: 2025

Inhaltsverzeichnis

1.	Projektüberblick	0
2.	Einleitung	1
3.	Mathematische Grundlagen großer Sprachmodelle	1
3.1.	Die Transformer-Architektur	1
3.2.	GPT-2 small	1
3.3.	Training großer Sprachmodelle	3
3.4.	Kausale und antikausale Inferenz	4
4.	Infrastruktur und Methodik	4
4.1.	Datengrundlage	4
4.2.	Tokenisierung	4
4.3.	Trainingsdurchführung	5
4.4.	Infrastruktur der HOST	5
5.	Vorgehen und Resultate	6
5.1.	Anfängliche Misserfolge <code>cf1</code> und <code>cf2</code>	6
5.2.	Training von <code>anticausal1</code> und <code>causal1</code>	6
5.3.	Statistische Signifikanzprüfung durch Zweistichproben- <i>t</i> -Test	7
5.4.	Qualitative Analyse durch Inferenzbeispiele	7
5.4.1.	Kochrezepte	7
5.4.2.	Biographische Artikel	8
5.5.	Finetuning	9
5.5.1.	Deutsche Gesetzestexte	9
5.5.2.	Plenarprotokolle des Deutschen Bundestags	9
6.	Visualisierung	10
6.1.	Token-Embedding	10
6.2.	Positions-Embedding	12
7.	Webanwendung	13
7.1.	Architektur	13
7.2.	Funktionen von Unterseiten (Auswahl)	13
7.2.1.	Unterseite <code>/token/[id]</code>	13
7.2.2.	Unterseite <code>/token/embedding-space</code>	13
7.2.3.	Unterseite <code>/chat</code>	13
8.	Fazit und Ausblick	14
9.	Danksagung	A
10.	Quellen- und Literaturverzeichnis	B

1. Projektüberblick

Im Projekt DEversAI untersuche ich, ob KI-Sprachmodelle besser funktionieren, wenn sie Texte vorwärts oder rückwärts verarbeiten. Dazu habe ich zwei KI-Modelle auf Deutsch trainiert: eines erzeugt Text vorwärts, das andere rückwärts. Ziel ist es, herauszufinden, ob Rückwärts-Modelle neue Möglichkeiten eröffnen und ob Erkenntnisse aus englischer Forschung im Deutschen gelten.

Die Ergebnisse sind vielversprechend. Das Vorwärts-Modell liefert präzisere Vorhersagen, aber das neue Rückwärts-Modell kann auch gute Texte vom Ende aus verfassen - so bei Kochrezepten, Gesetzen und Bundestagsreden. Eigene komplexe Visualisierungen der Modellstrukturen zeigen, dass beide sprachliche Muster lernen, aber sich in Aufbau und Ausgabe unterscheiden.

In der entwickelten interaktiven Webanwendung kann die KI ausprobiert und getestet werden. Die Resultate belegen, dass Textverarbeitungsrichtung einen wesentlichen Einfluss auf die Leistungsfähigkeit von KI in der Sprachverarbeitung hat.

„What I cannot create, I do not understand.“

— Richard Feynman

2. Einleitung

Seit der Einführung des Transformer-Modells[1] sind große Sprachmodelle (*large language models*, LLMs) Vorzeige- und zugleich wichtigster Untersuchungsgegenstand der angewandten KI-Forschung[2]. Dabei haben sich autoregressive kausale Modelle auf Basis von Decoder-Only-Transformern wie GPT in der Praxis trotz theoretischer Schwachstellen[3] gegenüber nicht-kausalen Modellen wie BERT durchgesetzt[4–6]. Eher geringe Betrachtung, darunter in [7], fanden dagegen *antikausale* Modelle, welche den Zeitschritt und damit die Direktionalität des Textes umkehren. Daher sollen in diesem Projekt kausale mit parametergleichen antikausalen Modellen verglichen und anhand verschiedener Metriken evaluiert werden.

Dabei wird die Projekt- und damit auch Modellsprache auf die deutsche Sprache beschränkt. So kann überprüft werden, ob Erkenntnisse aus der angloamerikanisch dominierten Forschung sprachübergreifend gelten. Zudem ist eine weiterführende Abgrenzung zu existierenden Modellansätzen und -untersuchungen möglich.

Der Hauptteil der schriftlichen Arbeit beginnt mit einer illustrierten Erklärung großer Sprachmodelle, einer Herleitung ihrer Parameteranzahl und einer groben Erklärung ihres Trainings (3.). Die die Methodik und Vorgehensweise der Projektausarbeitung wird beschrieben (4.), ehe die konkreten trainierten Modelle vorgestellt und quantitativ sowie qualitativ und exemplarisch, einhergehend mit einer Untersuchung sprachlicher Eigenschaften der Korpora selbst evaluiert werden (5.). Nach ausführlicher Visualisierung latenter Räume der beiden Hauptmodelle (6.) wird die die Webanwendung kurz vorgestellt (7.). Abschließend werden die Ergebnisse, insbesondere die wesentlichen korpuspezifischen Unterschiede der direktionalkomplementären Modelle, zusammengefasst und ein Ausblick geplanter Weiterentwicklungen präsentiert (8.).

3. Mathematische Grundlagen großer Sprachmodelle

3.1. Die Transformer-Architektur

Die heute vorherrschende Architektur zur Definition großer Sprachmodelle ist die 2017 in [1] eingeführte Transformer-Architektur, welche rekurrente und konvolutionale Netzwerke für die Texterzeugung ablöste, indem nur noch auf den Attention-Mechanismus gesetzt wird. Abb. 1 zeigt die Originaldarstellung des Modells, welche bereits die duale Encoder-Decoder-Struktur aufzeigt.

Aufgrund der Fülle an öffentlich verfügbaren Informationen und Erklärungen dieser Modelle^[1] und der Seitenbegrenzung geschuldet wird auf eine ausführliche Erklärung des allgemeinen Modells verzichtet und stattdessen das konkrete GPT-2 small-Modell, welches als handhabbares und dennoch leistungsfähiges Modell gewählt wurde, im Folgenden präsentiert.

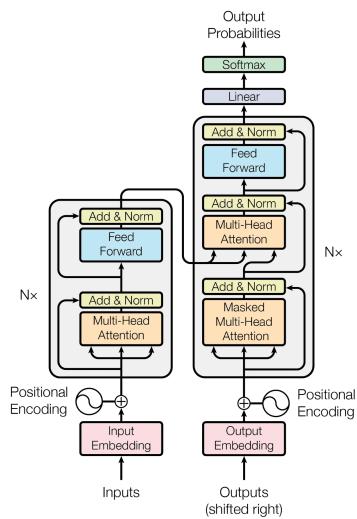


Abb. 1: Das in [1] gezeigte architekturelle Modell, Abb. übernommen.

3.2. GPT-2 small

Die von OpenAI 2019 vorgestellte GPT-2-Modellfamilie[11] beruht ebenso wie das Vorgängermodell GPT auf einer Decoder-Only-Architektur, sodass der Enkodierungsteil des Netzwerks nicht implementiert wird. Die Verarbeitung erfolgt in drei Schritten:

^[1]Zu erwähnen seien neben den einleitenden Fachartikeln insbesondere [8], [9] und [10].

1. Einbettung der Tokens in den Embedding-Raum,
2. Mehrfache Verarbeitung der Embedding-Vektoren in Transformer-Blöcken, sodass die Vektoren über Self-Attention kontextualisiert werden sowie
3. Abbildung auf Wahrscheinlichkeitsdistribution über Tokens.

Hyperparameter des Modells sind dabei die Embeddingdimensionalität E , Kontextgröße P , Vokabulargröße V , Anzahl von Transformerblöcken n_{layer} und Anzahl von Attention-Heads pro Block n_{head} . Abweichend von der ursprünglichen GPT-2-Architektur wird in hier trainierten Modellen in allen `LayerNorm`- und `Linear`-Schichten kein Bias verwendet, da dieser sich leicht negativ auf das Training auswirkt[12]. In Abb. 2 wird ein grober Überblick des Datenflusses zur Inferenzzeit dargestellt.

Am Anfang der Datenverarbeitung steht eine Einbettung der Tokens in den latenten E -dimensionalen Embedding-Raum des Modells. Dieser erfolgt über eine Parametermatrix der Größe $V \times E$. Durch das positionale Embedding, einer lernbaren $P \times E$ -Lookup-Tabelle, deren Ergebnisse im Anschluss mit den Vektoren aus E aufaddiert werden, kann der Embedding-Vektor mit der enkodierten Positionsinformation augmentiert werden^[2].

Jeder der folgenden n_{layer} Decoder-Transformer-Blöcke ist nun identisch aufgebaut (s. Abb. 3). Zu Beginn steht eine `LayerNorm`-Schicht mit E Parametern, welche entlang jeder Embedding-Dimension die Distribution der Elemente des Eingangstensors auf $\mu = 0, \sigma = 1$ normalisiert und eine lineare Transformation ausführt. Sie stabilisiert den Trainingsprozess und beschleunigt die Generalisierung[14,15].

Als definierendes Element folgt darauf eine Masked Multihead Self Attention-Schicht (s. Abb. 5). Sie besteht aus zwei Unterschichten mit lernbaren Parametern, die beide lineare Transformationen beschreiben. Die erste erzeugt aus den Eingaben die drei für Attention benötigten *Query*, *Key*- und *Value*-Matrizen^[3]. Sie wird als zu multiplizierende $E \times 3E$ -Matrix gespeichert. Die zweite parametrisierte Unterschicht bildet die durch Matrixmultiplikation berechneten Attention-Werte zurück auf Embedding-Vektoren ab und wird als $E \times E$ -Matrix gespeichert.

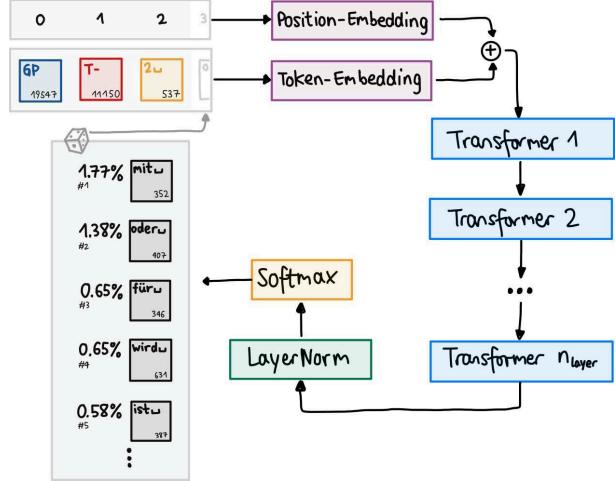


Abb. 2: Aufbau des GPT-2-Modells am Beispiel der Kausalinferenz mit Eingabe `GPT-2`.

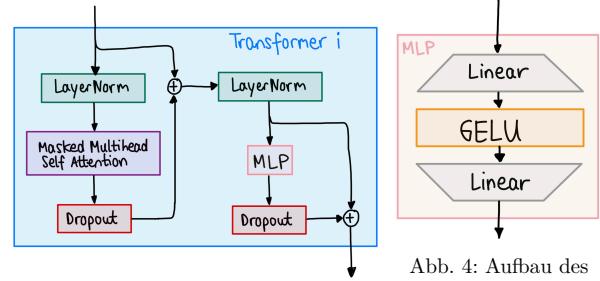


Abb. 3: Die Bestandteile eines Transformer-Blöcks des GPT-2-Modells.

Abb. 4: Aufbau des Multi-Layer Perzepton.

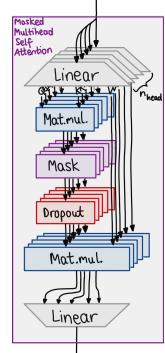


Abb. 5:
Schematischer Aufbau der Masked Multihead Self Attention.

^[2]Im ursprünglichen Transformer[1] wurde diese Enkodierung statt durch lernbare Parameter als unterschiedliche sinusoidale Funktionen fixiert; seit [13] gilt dies jedoch als überholt.

^[3]Auf eine genaue Beschreibung des Attention-Mechanismus wird hier ebenfalls der Kürze halber verzichtet, es sei auf die zitierte Literatur verwiesen.

Zudem liegt eine Dropout-Schicht vor, welche mit einer zuvor festgelegten Wahrscheinlichkeit p_{drop} einzelne Neuronen deaktiviert, um Überanpassung an Trainingsdaten zu vermeiden[16,17].

Die Ausgaben des Attention-Blocks werden nach erneutem Dropout zu den ursprünglichen Werten addiert – dieser Schritt verhindert katastrophales Vergessen der Eingabewerte – und nach weiterer LayerNorm mit E Parametern dienen die Resultate als Eingabe für eine MLP-Schicht, welches eine versteckte Schicht mit $4E$ Neuronen enthält und dann zurück auf E abbildet. Dieses Feedforward-Netzwerk hat insgesamt $8E^2$ Parameter: je $4E^2$ Gewichte zwischen zwei Schichten.

Insgesamt hat jeder der n_{layer} Transformerblöcke $E + (3E^2 + E^2) + E + (8E^2) = 12E^2 + 2E$ Parameter. Zusammen mit den anfänglichen Embeddings und einer finalen bias-freien LayerNorm ergibt sich $(V + P)E + n_{\text{layer}}(12E^2 + 2E) + E$ als Gesamtparameterzahl. Für die verwendeten Hyperparameter, die bis auf eine in Abschnitt 4.2 erwähnte Ausnahme denen von GPT-2 small entsprechen (s. Abb. 6), ergibt sich folglich[18]:

$$\begin{aligned} & (50'304 + 1024) \cdot 768 + 12 \cdot (12 \cdot 768^2 + 2 \cdot 768) + 768 \\ &= 50'304 \cdot 768 + 1024 \cdot 768 + 12 \cdot 7'079'424 + 768 \\ &= 38'633'472 + 786'432 + 84'953'088 + 768 \\ &= 124'373'760 \end{aligned}$$

V	50304
E	768
P	1024
n_{layer}	12
n_{head}	12

Abb. 6: Hyperparameter des Modells.

Somit hat das Modell etwa 124.37M zu trainierende Parameter.

3.3. Training großer Sprachmodelle

GPT-2 ist ein autoregressives LLM, was bedeutet, dass ein Inferenzschritt im kausalen Fall dem Vorhersagen des nächsten Tokens nach einer Liste gegebener Tokens, dem Kontextfenster, entspricht. Um längeren Text erzeugen zu können, wird dementsprechend das Kontextfenster mit dem neuen Token konkateniert und die Inferenz erneut ausgeführt. Das Training erfolgt daher über überwachtes Lernen, indem eine große Anzahl an Beispielen von Kontextfenstern und folgenden Tokens bereitgestellt wird, anhand derer das Modell Syntax, Grammatik, Struktur und Bedeutung der bereitgestellten Sprache erlernen soll.

Im konkreten Prozess werden die Parameter des Modells dabei zu Beginn zufällig initialisiert. Es beginnt die Trainingsschleife, in welcher für eine festgelegte Anzahl an Iterationen jeweils ein Trainingsschritt ausgeführt wird. Dieser besteht darin, dass das Modell für eine Anzahl an Beispielen (der Batchgröße) das nächste Token vorhersagt. Da das vorliegende Token bekannt ist, kann dessen Wahrscheinlichkeit der finalen Schicht entnommen werden. Häufig wird der negative natürliche Logarithmus, (*negative log likelihood*, NLL) dieses Werts als Verlustfunktion gewählt[19] und ist im Laufe des Trainings über alle Trainingsbeispiele zu minimieren. So wird die Wahrscheinlichkeit, das vorliegende Token auszugeben, im Trainingsverlauf maximiert.

Zu diesem Zweck wird über den Backpropagation-Algorithmus der Gradient aller Parameter bezüglich des Verlusts berechnet, damit diese über einen geeigneten Optimizer im Anschluss angepasst werden können. In der Praxis bewährt hat sich AdamW[20,21], eine Verbesserung von Adam[22], einem adaptiven Optimizer, welcher neben dem Hyperparameter der Lernrate η jedem Parameter basierend auf dessen Gradientenverlauf eine individuelle Lernrate zuweist.

Die Lernrate selbst bleibt dabei nicht konstant, sondern wird am Anfang linear erhöht (Warmup), um dann über Kosinus-Annealing[23] im Trainingsverlauf reduziert zu werden. So wird eine Überanpassung in den ersten Schritten verhindert, im Anschluss ein schnelles Training

gestattet und zum Schluss Fluktuation vermieden. Um Anpassungen auf Basis zufälligerweise nicht repräsentativer Trainingsbeispiele sowie die klassischen Probleme der explodierenden Gradienten[24,25] zu verhindern, wird zudem Gradient Clipping[26] genutzt, sodass die Anpassung der Parameter pro Schritt zusätzlich limitiert wird. Weight Decay[27,28] verbessert ebenfalls Generalisierung durch Bestrafung von Parametern zu hohen Beträgen.

3.4. Kausale und antikausale Inferenz

Der Einsatz eines großen Sprachmodells nach dem beschriebenen Training zur Textgeneration wird als Inferenz bezeichnet. Die Inferenz eines autoregressiven Modells auf Basis einer Eingabe läuft wie folgt ab: die Eingabe wird (als Liste von Tokens) ins Kontextfenster des Modells gelegt, um die Wahrscheinlichkeitsdistribution eines weiteren Tokens mittels der Modellschichten zu berechnen. Aus dieser wird ein Token stochastisch gewählt, ins Kontextfenster hinzugefügt und der Prozess wiederholt, bis eine suffiziente Anzahl an Tokens generiert wurde.

Alle laut [29] relevanten LLMs basieren dabei auf der natürlich erscheinenden[7] kausalen Inferenz. Dabei wird basierend auf dem Kontextfenster stets das folgende Token vorhergesagt und so der Text in Leserichtung vervollständigt. Im Gegensatz dazu steht die antikausale Inferenz, welche entgegen der Leserichtung arbeitet. Beispielhaft illustriert werden beide in Abb. 7.

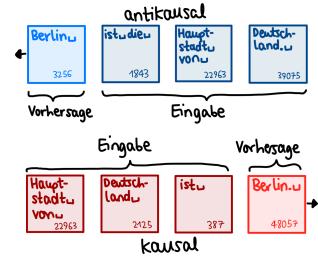


Abb. 7: Beispiele der Direktionalität.

4. Infrastruktur und Methodik

4.1. Datengrundlage

Noch vor dem Training steht die Wahl einer unter den Gesichtspunkten der Verfügbarkeit und Qualität geeigneten Datengrundlage. Hauptdatensatz ist daher das deutsche Fragment von OSCAR[30], ein auf Common Crawl beruhender sprachlich aufgeteilter Korpus, ergänzt um den Gesamtkorpus der deutschen Wikipedia[31]. Die Wikipedia-Daten wurden mithilfe von [32] aus XML-Datenquellen in ein JSON-Format konvertiert, während OSCAR bereits in einem solchen Format bereitgestellt wird. Ein eigens entwickeltes Rust-Programm dient dann der Destrukturierung in ein metadatenfreies Textformat, sodass die gesamte Datenmenge rein textuell vorliegt.

4.2. Tokenisierung

Dieser Datensatz wurde als Grundlage für einen in Rust programmierten nebenläufigen Tokenizer auf der Grundlage von Byte Pair Encoding (BPE)[33] verwendet, um insgesamt 50000 zusammengesetzte Tokens zu generieren. Dabei wurden zwei korpusentsprechende Anpassungen vorgenommen:

1. Der Text wird nicht nach Wortgrenzen voraufgeteilt, es findet also keine Pre-Tokenization statt. Dies gewährleistet, dass auch mehrwortige Tokens (wie `in der`) gebildet werden können, und stellt sicher, dass kein Sonderzeichen eingesetzt werden muss, um Tokens mit Leerzeichen repräsentieren zu können.^[4]
2. Statt in jedem Iterationsschritt nur das häufigste Tokendigramm ((l_1, r_1)) zu einem neuen Token $l_1 \circ r_1$ zusammenzufügen, wird ein Parameter $\eta \in [0, 1]$ eingeführt. Hat (l_1, r_1) eine bestimmte Häufigkeit f , so werden auch alle weiteren Tokens (l_n, r_n) hinzugefügt, für deren

^[4]Im ursprünglichen GPT-2-Vokabular wird hier `Ġ`, `LATIN CAPITAL LETTER G WITH DOT ABOVE`, genutzt[34].

Häufigkeit $f_n \geq \eta \cdot f_1$ gilt und die nicht durch ein bisheriges Token bereits aufgelöst werden können, also $\forall i \in [1, n - 1] : l_n \neq r_i \wedge r_n \neq l_i$.

Im Randfall $\eta = 1$ ergibt sich das typische BPE-Verfahren, während für sinkende Werte von η mehr Tokenpaare pro Schritt zusammengefügt werden. Heuristisch wurde für das Training der stückweise lineare η -Scheduler $\eta : [0, 1] \rightarrow [0, 1]$, $\eta(t) \mapsto \begin{cases} 0.8 & \text{falls } t \leq 0.3 \\ 0.8 + \frac{t}{7} & \text{falls } t > 0.3 \end{cases}$ genutzt, wobei $t \in [0, 1]$ den Anteil bereits ersteller Tokens beschreibt.

Zu den 50000 zusammengesetzten hinzu kommen die $2^8 = 256$ anfänglichen Tokens jedes möglichen ursprünglichen Bytewertes, die als Basistokens bezeichnet werden. Die Gesamtvokabulargröße ist somit $50000 + 256 = 50256$ ^[5]. Abb. 8 zeigt einen Auszug aus diesem Vokabular, wobei \square und \square_L ein enthaltenes Leerzeichen respektive einen Absatz repräsentieren.

ID	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275
Token	en	er	e	ch	en	ei	un	er	t	s	n	,	st	an	.	d	ie	el	in	sch
ID	5000		10000		15000		20000		25000		30000		35000		40000		45000		50000	
Token	pflicht	Zimmer	angemeldet	visitor	Balkon	aul	shiel	gigan	Verlangen	Obama										
ID	50246		50247		50248		50249		50250		50251		50252		50253		50254		50255	
Token	169	N Nun	weisen,	1990),	fluor	Dekaden	taucht	Backup	Zurzeit	rätsel										

Abb. 8: Eine Auswahl zusammengefügter Tokens aus dem deutschen Vokabular.

Der gesamte Korpus wurde über einen Zeitraum von zirka drei CPU-Monaten in die Tokenrepräsentation enkodiert und als Binärdateien, jeweils von Bytepaaren (Big Endian), gespeichert.

4.3. Trainingsdurchführung

Der Trainingscode beruht auf Karpathys nanogpt[12], welcher dem bereits erwähnten Aufbau des GPT-2-Modells von OpenAI vollständig folgt und das Modell in PyTorch[35] implementiert. Vollständig neu wurde nur die Logik zum Laden der Datenfragmente sowie die zur CLI-interaktiven Evaluation geschrieben; die Modelldefinition wurde zielgemäß beibehalten.

Anzumerken ist, dass es zum Training des antikausalen Modells keiner Anpassung der Modellstruktur bedarf, da es genügt (s. Abb. 9), die Tokens der Eingabedatei umzukehren und Modelleingabe X mit erwünschter Vorhersage Y zu tauschen. Durch diese Umkehr der Tokenzeit agiert die kausale Attention-Maske antikausal.

	Eingabe	Ausgabe
kausal	i	$i + 1$
	$i + 1$	$i + 2$
	\dots	\dots
antikausal	$i + P - 1$	$i + P$
	$i + P$	$i + P - 1$
	$i + P - 1$	$i + P$
	\dots	\dots
	$i + 1$	i

Abb. 9: Indexintervalle der Trainingsbeispiele ab i .

Zum Zwecke von Training und Inferenz wurde durch die Hochschule Stralsund (HOST) das GPU-Cluster LLM-HOSTed („Kira“) dankenswerterweise bereitgestellt. Es handelt sich bei den in diesem Projekt verwendeten um die ersten auf den GPUs dieses Clusters[36] trainierten Sprachmodelle. Zur Verfügung stehen insgesamt zehn NVIDIA A100-GPUs, von denen jedoch zwei dauerhaft und vier regelmäßig beansprucht werden, sodass das mittelfristige Training der LLMs auf vier GPUs mittels DDP^[6] parallelisiert ausgeführt werden kann.

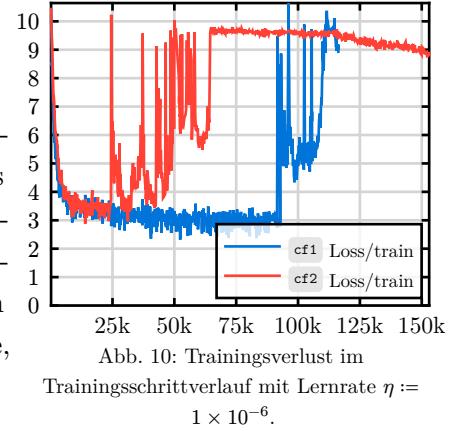
^[5]Typischerweise wird noch ein End-Of-Text (EOT)-Token ergänzt, sodass die Vokabulargröße 50257 beträgt; da der zu encodierende Text in diesem Fall jedoch auf UTF-8 beschränkt ist, kann das ansonsten nicht auftauchende Basistoken `0xff` = 255 diesen Zweck erfüllen.

^[6]Kurz für *distributed data parallel*[37], eine Methode zur Parallelisierung des Modelltrainings auf mehrere GPUs in PyTorch, mittlerweile für größere Modelle teils überholt[38].

5. Vorgehen und Resultate

5.1. Anfängliche Misserfolge `cf1` und `cf2`

Die ersten Versuche des Trainings eines kausalen LLM begannen Anfang Dezember 2024, nachdem das Vokabular bereits erfolgreich trainiert und der Korpus über diese Tokens enkodiert worden war. Abb. 10 zeigt das Verhalten der Verlustfunktion über den Zeitraum der Trainingsschritte für die ersten beiden Anläufe. Die x-Achse zeigt dabei die Trainingsschritte, die y-Achse den NLL-Verlust.



Es ist zu erkennen, dass der Verlust erwartungsgemäß innerhalb der ersten 5k Schritte beständig sinkt und der Betrag der Steigung bis Schritt 25k stark sinkt.

Ab Schritt 90k für `cf1` (bzw. 25k für `cf2`) ist allerdings zu erkennen, dass der Wert der Verlustfunktion schnell ansteigt, zeitweise stark fluktuiert und dann vergleichbar mit dem anfänglichen Wert zwischen 9 und 10 NLL stagniert, was insbesondere für `cf2` erkennbar wird. Die Modelle verlernen demzufolge die Fähigkeit der Textinferenz vollständig; die Ausgabe ist eine inkohärente Tokenkette, die nicht von einem zufälligen Modell zu unterscheiden ist.

Der Grund für dieses Verhalten konnte bisher nicht eindeutig identifiziert werden. In folgenden Trainingsverläufen wurde der Wert des Gradienten genauer überprüft, jedoch konnte keine Anomalie festgestellt werden. Der Versuch, das Modell von einem bisherigen Checkpoint wiederherzustellen, scheiterte ebenfalls: bei weiterführendem Training von `cf1` ab Schritt 90k, 80k und 50k mit bis auf Seed gleichen Hyperparametern konnte ein vergleichbarer hoher Anstieg des Verlusts beobachtet werden.

5.2. Training von `anticausal1` und `causal1`

Durch Ablationstests der relevanten verwendeten Hyperparameter wurde die Lernrate als für den zu verhindernenden zuvor festgestellten Verlustanstieg entscheidenden Faktor bestimmt. Eine sechsfache Erhöhung der Lernrate auf 6×10^{-6} (nach Anwendung des Gradient Clipping wie bisher) führte zwar zu höheren lokalen Fluktuationen, insgesamt aber zu stabilerem Training und einer schnelleren Modellkonvergenz. Das Training erfolgte hier bis Schritt

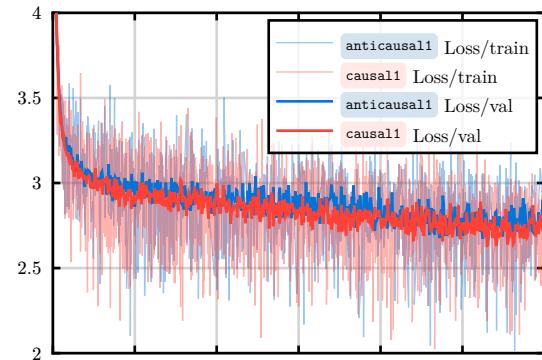


Abbildung 11: Trainingsverlust der kausalen und antikausalen Modelle mit Lernrate $\eta := 6 \times 10^{-6}$.

300k für das kausale und antikausale Modell. Abb. 11 zeigt den Wert der Verlustfunktion für Trainings- und Validierungsdaten im Trainingssverlauf, zur besseren Veranschaulichung skaliert auf $2 \leq y \leq 4$. Die so trainierten Modelle fungieren als Basismodelle (*foundational model*) und werden respektive als `causal1` und `anticausal1` bezeichnet.

Modell \ Shard	OSCAR			Wikipedia		
	1	2	3	1	2	3
anti causal1	2.8125	2.7818	2.8352	3.0861	3.0894	3.0876
	2.7183	2.6903	2.7417	3.0107	3.0136	3.0124
	2.6442	2.6174	2.6670	2.9638	2.9667	2.9656
causal1	2.7850	2.7548	2.8083	3.0559	3.0591	3.0577
	2.6877	2.6589	2.7110	2.9631	2.9662	2.9645
	2.6218	2.5941	2.6444	2.9166	2.9197	2.9182

Abbildung 12: Verlust auf Validierungsdaten.

Abb. 12 zeigt den Verlust ausgewählter Checkpoints auf nicht zum Training verwendeter Validierungsdaten. Der Verteilung entsprechend ist eine geringere Anpassung an den Wikipedia-

Korpus erkennbar. Zudem weist `causal1` zu allen ausgewählten Trainingszeitpunkten und für alle Validierungsshards einen leicht geringeren Verlust im Vergleich zu `anticausal1` auf, was eine höhere Generalisierungsfähigkeit kausaler Modelle indiziert.

5.3. Statistische Signifikanzprüfung durch Zweistichproben-*t*-Test

Für die konkreten trainierten Modelle sind diese Unterschiede statistisch signifikant, wie ein Zweistichproben-*t*-Test[39] in SciPy[40] belegen konnte, für den zuvor $\alpha = 0.05$ [41] festgelegt worden war. Für $H_0 := \text{"Verlust}_{\text{anticausal1}} = \text{Verlust}_{\text{causal1}}$ " und je 200k gemessenen Batches pro Korpus, OSCAR bzw. Wikipedia, liegt die Wahrscheinlichkeit der Observationen unter Annahme von H_0 (*p*-Wert[42]) bei 4.78×10^{-8} bzw. 4.32×10^{-145} . In beiden Fällen gilt daher $p < \alpha$ und H_0 wird verworfen. Die Effektstärke d nach Cohen[43] liegt bei 0.017 bzw. 0.081.

5.4. Qualitative Analyse durch Inferenzbeispiele

Um das Grundmodell testen und mit anderen kleinen LLMs vergleichen zu können, können neben den quantitativen Analysen auch qualitative Überprüfungen zu Rate gezogen werden. Hierfür wurden mit Kochrezepten und biographischen Artikeln zwei Beispiele für typische Textstrukturen gewählt.

5.4.1. Kochrezepte

Abb. 13 zeigt für den Anfang eines typischen Rezepts vier beispielhafte Ausgaben.

Prompt: Rezept: Griechischer Salat Zutaten: n			
2 TL Curry mit getrocknetem Thymian, fein gehackt , 40 g Wasser, 1 TL Salz , 1 Prise Salz Zutaten: 1 rote Zwiebel 2 EL Knoblauchzehen, fein gehackt , 2 EL Olivenöl Zubereitung: Die Eier trennen, die ungeschälten Eigelbe mit dem Zucker einschlagen und zum Mehl hineingeben. Mehl sieben und cremig schlagen. Die Eigelbe und die Knoblauchzehen dazugeben und alles vermengen. Mit einer Gabel oder mit einem Messer in ca. 5 cm dicke Scheiben schneiden. Die restlichen Eigelbe mit Salz steif schlagen. Die Zwiebel schälen, längs halbieren und in etwas Butter weich dünsten. Alles zusammen mit dem Curry und Knoblauch dazugeben und gut mitrösten. Kurz aufkochen lassen und für ca. 75 Minuten bei schwacher Hitze köcheln lassen. Alles nochmals durchschlagen, bis das Gemüse komplett kochend ist. Nach oben 10	50 g Wasser 50 g Zucker 60 g Olivenöl 1 EL Sojasauce ½ TL Muskatnuss Salz Pfeffer Zubereitung: Die Eier trennen. Den Zucker einrieseln lassen. Den Saft aussprechen. Den Eischnee einröhren. Die eingeweichten Sojasauce einröhren. Die Grießbutter mit dem Salz steif schlagen. Anschließend die Magerquark-Frischkäse-Masse dazugeben. Alles zu einer glatten Creme kneten. Den Teig für mindestens 12 min. gehen lassen. Eventuell noch etwas Butter oder etwas Milch beigeben. Den Teig nochmals für mindestens 12 min. gehen lassen. Als Nachspeisen servieren (meist mit Butterbrot oder Käseformfüllung). Nährwertangaben Die Nährwertangaben sind ohne	400 g Gemüse (je nach Größe) 350 g Champignons mit Saft 3 EL Olivenöl 2 EL Butterschmalz 2 TL Meersalz 2 EL Zitronensaft Anleitung: Champignons abwaschen, entkernen und in Scheiben schneiden. Alle Zutaten in eine Schüssel geben, sodass alle Zutaten gemeinsam sind. Vorsichtig pürieren und zu einem geschmeidigen Dressing verarbeiten. Nun ebenfalls das Mehl (Margarine) mit in die Schüssel geben und alles gut miteinander vermengen. Bestreichen und mit Salz und Pfeffer abschmecken. Diese Pfanne sollte man wirklich nur empfehlen, da das Öl auf dem Öl direkt angebrannt ist. Natürlich kann man den Boden damit ausdämpfen und es ist ein Problem mit dem Fett. Für die Zubereitung am besten mit Backpapier auslegen. Mit dem Zitronensaft ablöschen und kurz abkochen. Gute Brotschüssel	Zutaten: 1 Ei 2 EL Olivenöl 5 EL Butter (3 EL Öl) 1 EL Zucker 2 EL Zitronensaft 2 rote Zwiebeln 1 EL Apfelessig 2 EL Weißweinessig Zutatenliste: Für die Tomatensuppe: 400 g Blattspinat (mittelgroß) 2 EL Weißweinessig 100 g geriebene Zwiebeln 50 g Petersilie 150 g Champignons 25 g Dill 1/2 EL Zitronensaft Salz & Pfeffer Den Spinat schälen und fein würfeln. Den Butter in einer Pfanne erhitzen und die Zwiebelwürfel darin platzen lassen. Die Knoblauchzehen schälen und fein hacken. Den Knoblauch fein hacken

Abb. 13: Vier beispielhafte Rezeptvorschläge von `causal1`.

Diese Ausgaben zeigen, dass das Basismodell dazu in der Lage ist, die Struktur und den Aufbau typischer deutscher Rezepte zu imitieren und plausible Ausgaben zu erzeugen, jedoch inkohärente Anweisungen und unpassende und willkürliche Zutaten ausgibt. Neben dem induzierten Schema eines Rezepts besteht kein wirklicher Zusammenhang zwischen Eingabe und Resultat. Allerdings sind alle Ergebnisse (mit einer Ausnahme) grammatisch und syntaktisch korrekt und Begriffe für Zutaten, Verarbeitungsmethoden und Verbindungswörter werden abgesehen von ihrer Bedeutung grundsätzlich auch fehlerfrei eingesetzt. Durchgeführte Vergleiche mit einem vortrainierten englischsprachigen GPT-2-small-Modell bestätigen ähnliche Ergebnisse.

Versucht man, das antikausale Modell auf identische Art und Weise zu testen, enttäuschen die Ergebnisse. Denn das Modell sagt den Text vorher, auf den das Rezept folgt – im Fall von Websites oft entweder ein weiteres unzusammenhängendes Rezept oder wie in Abb. 14 repetitiver regulatorischer Text am Ende des bisherigen Trainingsbeispiels^[7]. Stattdessen sollte wie in Abb. 15 ein am Ende

Informationen allein in der Regel nicht dazu verwendet werden, dich zu identifizieren.
Marketing Marketing
Die technische Speicherung oder der Zugriff ist erforderlich, um Nutzerprofile zu erstellen, um Werbung zu versenden oder um den Nutzer auf einer Website oder über mehrere Websites hinweg zu ähnlichen Marketingzwecken zu verfolgen.

Rezept: Griechischer Salat

Zutaten:

Abb. 14: Fehlerhaftes Prompting `anticausal1`.

des gewünschten zu erzeugenden Textes stehender Teil übergeben werden, um eine passende Generation zu gewährleisten.

<p>EISCHNEE SCHOKOLADE KÜSCHEL</p> <p>Eischnee schälen und in kleine Stücke schneiden. Petersilie fein hacken und zusammen mit dem Eigelb in die Masse geben.</p> <p>Zucker/Zucker Verarbeiten</p> <p>Die zerdrückte Schokolade mit dem Zucker 3 Minuten kochen lassen. Die Schokolade leicht abkühlen lassen. Die Schokolade abkühlen lassen. In Gläser abfüllen und für 1 Stunde in den Gefrierschrank stellen. Die Himbeeren vorher fein püriert haben.</p> <p>Das Eiweiß steif schlagen und die Zartbitterschokolade unterheben. Mit 200 ml kaltem Wasser ablöschen. Den Puderzucker unterheben und den Saft beiseiteten.</p> <p>Die Erdbeeren und etwas abgeriebene Zartbitterschokolade hinzufügen. Die gehäuteten Pinienkerne und den restlichen Zitronensaft hinzufügen. Die Zartbitterschokolade abkühlen lassen. Die Zartbitterschokolade zugeben. Mit dem Kakao bestreuen. Das Erdbeercompott darüber verteilen.</p>	<p>KRÖPFLER MIT REIS-SOJA-MASSE</p> <p>vorgeheizten Backofen bei 200 °C für ca. 1 Stunde für ca. 20 Minuten rasten lassen. Die Fritteuse damit auskleiden und im vorgeheizten Backofen bei 180 °C für ca. 20 Minuten backen.</p> <p>Herausnehmen und abkühlen lassen.</p> <p>Die Mascarpone waschen und mit einem Messer in Spalten schneiden.</p> <p>Die Mascarpone waschen und halbieren. Dünne Haut abschneiden. Die Mascarpone waschen, abtrocknen und in ca. 1 cm dicke Scheiben schneiden. Den Strunk abschneiden und dann in ca. 1 cm dicke Scheiben schneiden. Die Mascarpone waschen, einen Topf mit Butter einfüllen und die harte, spröde Haut abschneiden.</p> <p>Die Mascarpone abschmelzen und mit einem Messer feine feine Streifen abschneiden. Das Gemüse waschen und in Stücke schneiden. Den Salat waschen und in Stücke schneiden. Die Nudeln zugeben. Dann die Zitronensauce darauf verteilen und alles aus der Pfanne servieren.</p>	<p>KÖCHEN HESSEN: AM BESTEN HEIS SERVIEREN. GUTEN APPETIT!</p> <p>Vorbereitung</p> <p>Kartoffeln Zutaten</p> <p>Sauerkraut, Blumenkohl, Spitzkohl, Brokkoli-Reis, Basmati-Reis, Walnusse, Rapsöl, getrocknete Sojabohnen, Gewürze, gemahlene Vanille, Zwiebeln, rote Linsen, Kichererbsen, Kämmel, Salz, Kräuter</p> <p>Pflanzliches Sauerkraut, Liebstöckel, Petersilie, Spargel, Stangenessellerie, Birne, Kürbis</p> <p>Zubereitung:</p> <p>Die Gewürze waschen und in 1,5 bis 2 cm dicke Scheiben schneiden. Kraut mit einem scharfen Messer längs halbieren. In einen Topf gießen und 5 Minuten sprudelnd köcheln. Gewürze zufügen, ins herausgekochte Wasser einröhren und die Gewürze hinzugeben, mit dem Stabmixer einröhren und weitere 5 Sekunden köcheln lassen.</p>	<p>GRÄSSE: NUDELN</p> <p>Jetzt ist Schluss mit Falafel.</p> <p>Zutaten:</p> <ul style="list-style-type: none"> 5 der Nudeln 5 Knoblauchzehen 25 g Salz 500 ml Öl 500 ml Pflanzenöl 500 ml Pflanzenöl 250 g Nudeln 500 ml Pflanzenöl 250 g Nudeln 500 ml Pflanzenöl Salz, Pfeffer <p>Zubereitung:</p> <p>Die Nudeln in heißem Öl in einer großen Pfanne von beiden Seiten zwei Minuten lang anbraten. Den Knoblauch dazugeben und alles schief anbraten.</p> <p>Mit 500 ml Pflanzenöl in einem Topf bei mittlerer Hitze 5 Minuten köcheln lassen. Den Knoblauch abziehen und den Quark unterrühren. Mit Salz und Pfeffer würzen. Mit Salz und Pfeffer würzen und durchziehen lassen.</p>
--	---	--	---

Prompt: Am besten heiß servieren. Guten Appetit!

Abb. 15: Vier beispielhafte Rezeptvorschläge von `anticausal1`.

Diese Beispiele legen ebenfalls ein Verständnis der globalen und lokalen Struktur, aber Unverständnis der konkreten Semantik des Rezepts nahe. Die Ausgaben scheinen im Vergleich repetitiver zu sein, ansonsten lernt das antikausale Modell ebenfalls den typischen Rezeptaufbau.

5.4.2. Biographische Artikel

Für die stochastiche Generation antikausaler biographischer Artikel ist ein häufiges gemeinsames Suffix dieser Textklasse nötig. Um das Modell dazu anzuleiten, dabei die Wikipedia-Trainingsdaten strukturell zu übernehmen, können dazu Fragmente wie die in Wikipedia-Artikeln enthaltenen Kategorie-Daten und Überschriften übernommen werden. Abb. 16 zeigt vier Artikel über Autorinnen, indem die Kategorien `Frau` und `Autor`^[8] genannt werden.

<p>PREIS FÜR DAS ERBE DES FEUERWEHRMANNES</p> <p>Preis - 3. Preis bei Jäkou, abgerufen am 17. Februar 2023</p> <p>Schriften</p> <p>Bücher</p> <ul style="list-style-type: none"> • Mónia Szabó: Aus dem Leben einer Frau. Ausgewählte Texte. Steidl, Göttingen 2009, ISBN 978-3-88243-573-2. <p>Als Autorin</p> <ul style="list-style-type: none"> • Mónia Szabó: Das Erbe des Feuers. Aus dem Englischen von Anja Fischer. Callwey, München 2009, ISBN 978-3-85967-802-6. • Mónia Szabó: Die Geschichte einer Frau. Ein Frauenessay, gelesen von Mónia Szabó, studiv., München 2011, ISBN 978-3-88074-534-5. • Mónia Szabó: Meine Frau ist kein Mann: Bilder die Fotografie von Frauen. Henrich & Henrich, Berlin 2014, ISBN 978-3-943967-57-3. • Mónia Szabó: Die Erinnerungen einer Frau. Übersetzt von Markus Schmid, Kiepenheuer & Witsch, Köln 2016, ISBN 978-3-462-01029-4. <p>Weblinks</p> <ul style="list-style-type: none"> • Das Leben einer Frau ist kontrastreich, siehe Literatur 	<p>Familie</p> <p>Alice Rejonoy ist die Autorin eines Fantasy-Romans. Sie ist mit dem englischen Autor Mark Chapman verheiratet. Ihre Tochter ist die Schauspielerin Jamie Chapman. Der Ehe entstammt ihre Tochter Rachelle Chapman, die ebenfalls als Schauspielerin tätig ist. Alice Rejonoy ist mit dem Schauspieler Mark Chapman verheiratet und hat zwei Kinder.</p> <p>Werke</p> <p>Seit 2018 schrieb Rejonoy einen Fantasy Roman („Deception“) des englischen Autors Richard Along. Ihr künstlerisches Schaffen befindet sich im Gatesway House in London.</p> <p>Bücher</p> <ul style="list-style-type: none"> • 2017. „Amaryllis“ in: „Amaryllis“ in: „The Bazaar of Kings“ von Malcolm Chapmans. Verlag BiblioBazaar, 2019. • „Amaryllis“ und „Red Line of Love“ von Malcolm Chapman. Verlag BiblioBazaar, 2015. <p>Weblinks</p> <ul style="list-style-type: none"> • Rejonoy bei dreamingchannel.com 	<p>Auf Tittelbach.tv, Abgerufen am 30. Januar 2013. In Deutschland wurde der Film am 29. Januar 2007 bei RTL ausgestrahlt und es wurden unter anderem die Filme <i>The Secret Life Of A Star</i> und <i>The Rockstar</i> gezeigt.</p> <p>„Girls For American Girl“: Der Animationsfilm steht kurz vor der Weltpremiere von „Girls For American Girl“ Star. In faz.net, 17. Januar 2013, Abgerufen am 18. Januar 2013. Am 9. Februar 2007 wurde ein Video zum Film veröffentlicht. Der Film wurde ohne den Namen „Girls For American Girl“ selbst gedreht.</p> <p>Girl For American Girl</p> <p>A Time To Love ist eine Serie mit einer Feuilleton-Darstellerin, die unter dem Titel A Time To Love von zwei Künstlerinnen mit dem Titel Girls For American Girl gedreht wurde, die erstmals im Ersten am 18. Januar 2013 ausgestrahlt wurden. A Time To Love“: Jetzt kommt die Serie ins Fernsehen. In: derwesten.de, 18. Januar 2013, Abgerufen am 12. Juli 2015. Seit 2006 wird das Video auf Englisch mit deutschen Untertiteln angeboten. <i>Dancing Girls: The Best of "Girl and Poppy"</i> auf last.fm</p> <p>Weblinks</p> <ul style="list-style-type: none"> • Video auf YouTube 	<p>Arbeit von Saverrias, zuerst als Stellvertreterin, später als Leiterin der Leitung und als Schatzmeisterin.</p> <p>Saverrias war von 1996 bis 2000 Vizepräsident und von 2006 bis 2010 Präsidentin. Von 2002 bis 2005 war Paula Saverrias Oppositionsführerin. 2005 wurde sie zur Präsidentin gewählt. Bei der Präsidentenwahl in Copacabana 2006 gewann Paula Saverrias mit 5,1 % der abgegebenen Stimmen. Bei der Präsidentenwahl in Kuala Lumpur 2016 wurde Paula Saverrias von Frieda Saverrias zur Nachfolgerin des nicht mehr kandidierenden Präsidenten gewählt.</p> <p>Die Wahl fand am 2. März 2017 statt. Leonore Saverrias, in: lavanguardia.com vom 4. März 2017 Auf Vorschlag der PCP setzte sie sich bei der Wahl gegen Aurelio Luria durch. Nach ihrem Wahlsieg von 2018 gab Saverrias ihre Rolle als Partei- und Religionsleader an. Saverrias ist Mitglied der Kommunistischen Partei Ecuatorian.</p> <p>Veröffentlichungen</p> <ul style="list-style-type: none"> • Jingdaka olaksin bangkla dakararalingayan triya. Modeshi, 2017 <p>Weblinks</p> <ul style="list-style-type: none"> • Englisch, englisch, indonesisch
--	--	---	---

Prompt: `N_N## Einzelnachweise N_N Kategorie:Frau Kategorie:Aut`

Abb. 16: Vier Endauszüge biographischer Artikel von `anticausal1`.

[7] Diese Art von Textfragmenten kommt auch in anderer Nutzung der Modelle vor und ist auf die Eigenschaften von Common Crawl zurückzuführen; weiterführende Filteransätze liegen jedoch außerhalb des Projektrahmens.

[8] Die Wikipedia-Kategorien nutzen das generische Maskulinum[44].

In drei der vier erzeugten Einträge halluziniert das Modell dabei die Biographie von Personen, die *Mónia Szabó*, *Alice Rejonoy* und *Paula Saverriás* genannt werden. Die Namensnennung ist konsistent und die Darstellung insgesamt stimmig, der zweite Text repräsentiert selbst die Familie bis auf Beruf des Ehemannes eindeutig. Text 4 enthält neben politischen Ereignissen auch eine scheinbare Veröffentlichung in indonesisch wirkenden Worten, die jedoch ebenfalls keine Bedeutung haben^[9]. Abb. 17 zeigt einen längeren biographischen Text eines amerikanischen Politikers.

5.5. Finetuning

Um die Generalisierungsfähigkeit der Modelle zu vergleichen und domänenspezifischere Texte erzeugen zu können, wurden beide Grundmodelle im überwachten Finetuning auf zwei verschiedene Datensätze trainiert. Das Finetuning verläuft dabei gleich zum bisherigen Training, jedoch wird die Lernrate η auf 6×10^{-5} fixiert und p_{drop} aufgrund des kleineren Datensatzes zur Verhinderung von Überanpassung auf 0.1 erhöht.

5.5.1. Deutsche Gesetzestexte

Der Gesamtkorpus der geltenden deutschen Bundesgesetzgebung wird vom Deutschen Bundestag in einem offiziellen Git-Repository[47] veröffentlicht. Die heruntergeladenen 170MB Markdown-Textdateien wurden in Rust tokenisiert, die Tokens dann im üblichen 90-10-Split auf 29.6M Trainings- und 3.3M Validierungstokens aufgeteilt. Der Trainingsverlauf wird in Abb. 18 gezeigt. Es ist zu erkennen, dass anfänglich aufgrund des Dropouts der Validierungsverlust stärker sinkt als der Trainingsverlust, etwa nach Schritt +800^[10] durch Overfitting der Validierungsverlust stagniert, während der Trainingsverlust weiter abnimmt.

5.5.2. Plenarprotokolle des Deutschen Bundestags

Ebenfalls vom Deutschen Bundestag herausgegeben werden die Plenarprotokolle aller Bundestagssitzungen seit 1949 im XML-Format[48]. Die 895MB umfassenden XML-Dateien wurden in ein Textdateiformat umstrukturiert und analog der Split in 170M Trainings- und 18.9M Validierungstokens ausgeführt. Das Training wurde bis auf Erhöhung der Trainingsschritte durch eine größere Datenmenge analog zu Abschnitt 5.5.1

^[9]Das Scheitern einer Übersetzung mit DeepL und Google Translate sowie die Aussagen von GPT-4o und pseudonymer Nutzer einer indonesischen Online-Community[45] legen dieses Ergebnis nahe. Da in Indonesien über 700 regionale Sprachen gesprochen werden[46], ist nicht ausschließbar, dass die Phrase in einer von ihnen einen gültigen Satz bildet; die Wahrscheinlichkeit, dass ein deutschsprachiges Sprachmodell die Grammatik einer dieser beherrscht, bleibt dagegen vernachlässigbar gering.

^[10]Die Schritte werden als zusätzlich zu den vorherigen gezählt, der absolute Trainingsschritt ist hier 300k + 800 = 300'800.

Anfang Mai 2008 bestand Pickardt aus einer im November 2007 verhängten Freiheitsstrafe von einem Jahr und drei Monaten. Pickardt gehörte zu einem der vom FCC beschuldigten Initiatoren eines Liberal-Technology-Korruptionsskandals, die ihre Vorwürfe gegenüber der Staatsanwaltschaft erklärten. Das FCC hatte später verlautbart, dass dieser „Ausbruch aus der Forensik“ Gegenstand mehrerer Indizien gewesen sei. Der FCC behauptet, dass ein Untersuchungsausschuss relevante Indizien ermittelt hatte. Der FCC habe Pickardt Ende April 2008 „im Kontext der relevanten Indizien und der verfügbaren Beweismaterialien“ befragt; er habe feststellen können, dass Berichte zur Veröffentlichung geführt haben. Der Untersuchungsausschuss stießte im Oktober 2008 die Ergebnisse der Untersuchung „aufgeklärter Fälle“ zu. Im Juni 2007 wurde Pickardt durch den FCC deshalb wegen versuchten Mordes angeklagt.

Öffentliche Äußerungen

Die Justiz in den USA und die Gefängnisbehörde der USA, welche zu mehrjährigen Haftstrafen verurteilt worden waren, wiesen ihre Vorwürfe zurück. Sie beantragten am 12. November 2007 beim Bundesgerichtshof in Washington, D.C. eine zehnjährige Freiheitsstrafe, gegen den er Berufung eingelegt hatte, die jedoch erst nach mehreren Prozessen eingeholt worden war. Ein Militägericht verurteilte ihn im Juni 2008 unter Berufung in den Aufsichtsrat zur Zahlung eines unentgeltlichen Zuschusses für bis zu 15 Jahre (21 Jahre) und eine Freiheitsstrafe. Die Parlamentskommission legte Berufung ein.

Weblinks

womenobserver.com

Einzelnachweise

Kategorie:Politiker (Vereinigte Staaten)
Kategorie:Mann

Abb. 17: `anticausal1`-generierte Biographie (Prompt von nun an *kursiv*)

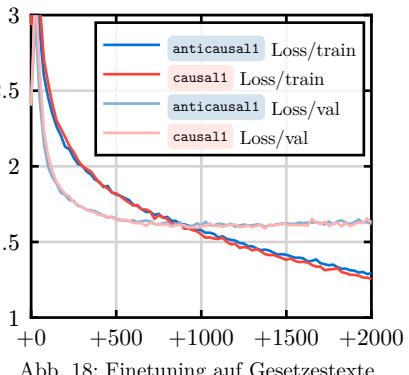


Abb. 18: Finetuning auf Gesetzestexte

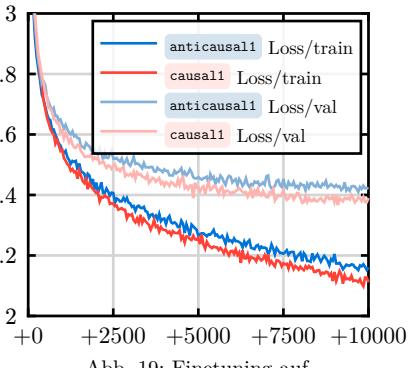


Abb. 19: Finetuning auf Plenarprotokolle

ausgeführt; Abb. 19 zeigt die Resultate, zu beachten seien die geänderte Skalen. Stärker als bei Gesetzestexten ist hier der Unterschied zwischen kausalem und antikausalem Modell zu erkennen, da der Verlust des ersten durchgehend geringer ist. Es lässt sich vermuten, dass die kausalen Argumentations- und Debattenstrukturen im Vergleich zu querverweisenden Gesetzestexten zeit- und damit direktonalitätssensitiver sind.

Abb. 20 und Abb. 21 zeigen zwei exemplarische Ausgaben der respektiven antikausalen Modelle. Das auf Plenarprotokollen trainierte Modell schafft es dabei, im Kontext des verwendeten Prompts (Investitionen) zu bleiben, jedoch nicht, die Bundeskanzlerin als Frau zu bezeichnen. Dies ist als wiedergegebene Verzerrung der Trainingsdaten erklärbar, da in nur vier (Merkel I-IV) der neunzehn Wahlperioden das Amt durch eine Frau bekleidet wurde.

6. Visualisierung

6.1. Token-Embedding

Wie bereits in Abschnitt 3.2 erwähnt, steht am Anfang der Datenpipeline des Transformers eine Token-Embedding-Layer, welche den als IDs repräsentierten Tokens über eine gelernte Lookup-Tabelle 768-dimensionale Vektoren zuweist, die in den folgenden Layern des Modells kontextualisiert werden. Bereits diese $\mathbb{R}^{50256 \times 768}$ -Matrix enthält semantische Informationen, die Aufschlüsse über die Funktion des Modells bieten können.

Um diese Daten anschaulich visualisieren zu können, wurde ein Python[49]-Programm geschrieben, um die Vektorliste mithilfe des Dimensionalitätsreduktionsalgorithmus PaCMAP[50] auf drei sowie zwei Dimensionen zu reduzieren. Der Algorithmus zeichnet sich durch die Fähigkeit aus, sowohl globale (wie PCA[51]) als auch lokale (wie t-SNE[52] oder UMAP[53]) Strukturen auf den niedrigdimensionalen Raum abbilden zu können[54].

Abb. 22 zeigt die Räume des direktonalkomplementären Modellpaars. Durch das Betrachten spezifischer Cluster dieser Räume können Informationen über die sprach- und korpus-spezifische Repräsentation der Tokens im Modell gewonnen werden. Diese können syntaktischer, grammatischer oder semantischer Natur sein: erstere enthalten die Klasse der Suffixcluster, welche insbesondere `causal1` nach der Eigenschaft partitionieren, mit einem Leerzeichen zu enden; zweitere umfassen unterschiedlich deklinierte Verben sowie Nomen unterschiedlicher Kasus, während als Beispiele für letztere die ebenfalls geometrisch segmentierbaren Cluster deutscher Vornamen, geografischer Bezeichnungen oder Temporalwörter zu

Transplantationsstelle nach § 22a des Transfusionsgesetzes, soweit die Regelungen der Daten in den Absätzen 3 bis 6 und 7 geregt werden oder die Daten der Vorschriften von einer Genehmigung nach § 23 des Transfusionsgesetzes ausgenommen sind oder die Daten für diese Stellen von den Vorschriften der Rechtsverordnung nach § 9 erfasst werden.

(4) Die Regelungen zu den Absätzen 5 und 7 sind in die hierfür einzubindenden Maßnahmen aufzunehmen.

§ 9 Beschaffung, Marktforschung, Bereitstellung von Medizinprodukten; Verordnungsermächtigung

(1) Eine Entscheidung des Bundesamtes für den Bevölkerungsschutz (Bundeshauptamt) für den Bereich der Versorgung der Bevölkerung kann von den §§ 3 und 5 genannten Anforderungen auch unter Berücksichtigung des Informationsbedürfnisses der Bevölkerung abweichen. Dabei dürfen die in den §§ 3 und 5 genannten Schwellenwerte nicht über- oder unterschritten werden.

(2) Betriebe, die Künstliche Intelligenz einsetzen, sind von § 2 betroffen.

Abb. 20: Antikausal generiertes Gesetz.

Es gibt keine Hemmnisse auf dem Weg zu besseren finanziellen Rahmenbedingungen für notwendige Investitionen. Der Prozeß der Osterweiterung muß wettgemacht werden und kann die Infrastruktur umstrukturiert werden.
(Beifall bei der SPD)

Eine zukunftsgerichtete Strukturpolitik würde gleichzeitig die soziale Lage in den neuen Bundesländern erheblich verbessern.
(Beifall bei der SPD)

In diesem Zusammenhang möchte ich auch den Versuch unterstützen, die neuen Ländern mit zu investieren und zu überziehen. Ich kann nur unterstreichen, was der Herr Bundeskanzler vorhin gesagt hat: Es wird zwar investiert, aber niemand wird bezweifeln, daß die begonnene Investition in neue Arbeitsplätze eine Chancen hat.
(Beifall bei Abgeordneten der SPD)

Liebe Frau Bundeskanzlerin, das Land braucht Investitionen in unsere Zukunft!

Abb. 21: Antikausale Plenardebatte.

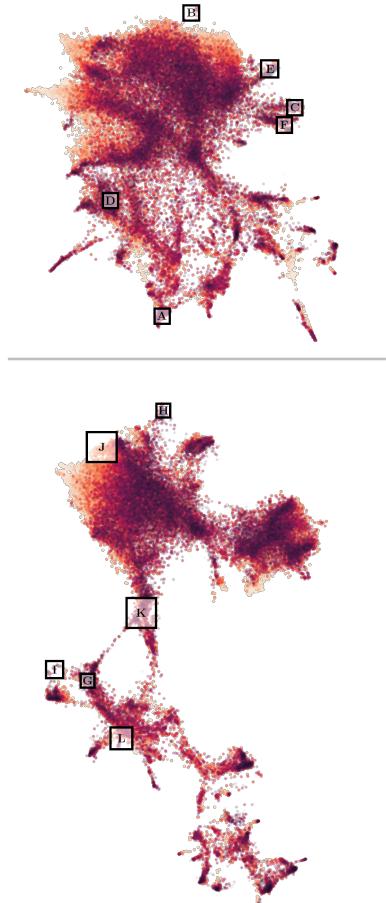


Abb. 22: Das 2D-Embedding von `anticausal1` (oben) und `causal1` (unten). Jeder der 50256 Punkte repräsentiert ein Token, eingefärbt nach dessen ID. Erstellt mit Pillow[55].

nennen sind. Abb. 23 zeigt für jedes Modell sechs beispielhafte und nennenswerte Cluster.



Abb. 23: (A-L) Zwölf Cluster, die ersten sechs (A-F) aus dem dimensionalitätsreduzierten latenten Embedding-Raum von `anticausal1`, die letzten sechs (G-L) von `causal1`. Erstellt mit Pillow[55].

6.2. Positions-Embedding

Während das Token-Embedding jedem Token einen Vektor im Embedding-Raum zuweist, assoziiert das Positions-Embedding jeder Position im Kontextfenster einen solchen. So können die Voraussagen des Modells von der absoluten Position der Tokens abhängen. Aufgrund des von Natur aus heuristischen Tokenisierungsprozesses und der häufigen Austauschbarkeit von Begriffen in natürlicher Sprache indiziert eine „Rauheit“ oder ein Rauschen eine fehlende Generalisierung des Modells auf die Daten. In Abb. 24 repräsentiert jede Kurve eine Dimension, die Abszisse die absolute Position des jeweiligen Tokens. Es ist zu erkennen, dass die Positions-Embeddings des kausalen Modells parameterinsensitiver sind als die des antikausalen.

Simpel quantifiziert werden kann dieses Maß durch die Bestimmung der „Rauheit“ über den durchschnittlichen Betrag der Abstände konsekutiver Komponenten (*mean absolute difference*, MAD). Über alle Dimensionen gemittelt erhält man für causal1 einen Wert von 1.068×10^{-3} , für anticausal1 6.745×10^{-4} , eine größere Abweichung gibt es bei den Ausreißern (siehe Abb. 25).

Abb. 26 zeigt den Verlauf des arithmetischen Mittels im Trainingsverlauf, wobei die Divergenz der Kurven nach etwa 175k Schritten erkennbar ist. Es bleibt uneindeutig, ob es sich hierbei um eine Anomalie im Trainingsprozess oder eine inhärente Eigenschaft antikausaler Modelle handelt; für letzteres spricht, dass auch das kausale Modell vom logistischen Regressionsmodell^[11] kurz vor Trainingsende abweicht.

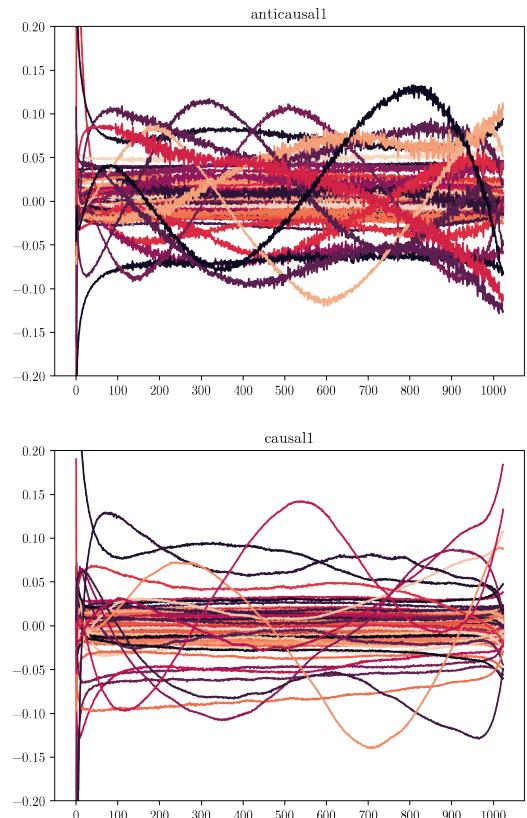


Abb. 24: Gelernte Werte des Positions-Embedding des Modellpaars, erstellt mit Matplotlib[56].

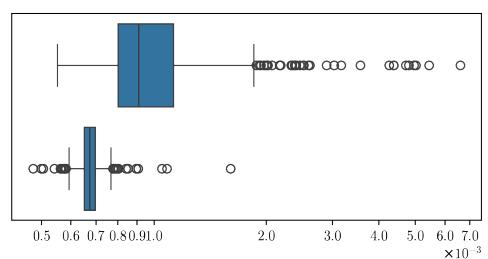


Abb. 25: Boxplot der MADs aller 768 Embedding-Dimensionen, log. skaliert, oben antikausal, unten kausal, erstellt mit Seaborn[57].

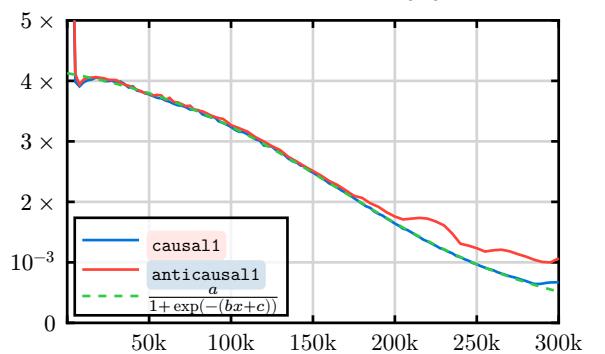


Abb. 26: Arithmetisches Mittel der MADs über alle Dimensionen im Laufe des Trainings samt logistischer Regression mit $R^2 = 0.9998$ im Intervall [50k, 250k]

^[11] $f(x) = \frac{a}{1 + \exp(-(bx+c))}, a = 4.478 \times 10^{-3}, b = -1.501 \times 10^{-5}, c = 2.466$

7. Webanwendung

Zum Zwecke der Nutzung und Visualisierung der in diesem Projekt trainierten Modelle wurde die interaktive Webanwendung [58] entwickelt. Der Quelltext[59] ist quelloffen und frei lizenziert.

7.1. Architektur

Der in der Webanwendung verwendete Technologiestack umfasst Svelte[60] samt SvelteKit[61] als Framework mit TypeScript[62,63] im Frontend. Zudem gibt es zum Zweck der Daten- und Verantwortungsseparation zwei in Python[49] mit FastAPI[64] geschriebene Backends, die im Code als `shallow-backend` und `deep-backend` bezeichnet werden. `shallow-backend` liegt dabei auf einem externen Linux-Server ([65]), während `deep-backend` auf dem hochschuleigenen Server ausgeführt wird. Reine Datenbankaufrufe wie die Abfrage von Korpusbeispielen für ein bestimmtes Token werden durch den `shallow-backend`-Webserver direkt beantwortet, während Inferenz und andere Modellanfragen über ihn relaisartig an `deep-backend` weitergeleitet werden. Zur bidirektionalen Kommunikation werden Websockets[66] verwendet, wobei das `shallow-backend` eine eigene Klientenverwaltung beinhaltet muss, um an jeden Client nur die angefragten Tokens zurückzusenden.

7.2. Funktionen von Unterseiten (Auswahl)

Es folgt eine kurze Beschreibung dreier Unterseiten der Webanwendung, welche die interaktiven Funktionen dieser exemplarisch illustrieren.

7.2.1. Unterseite `/token/[id]`

Diese Seite enthält Informationen zum Token der ID `id`. Diese visualisiert wie in Abb. 27 mithilfe von d3.js[67] den Tokenstammbaum, d.h., die vollständige Ableitung bis hin zu Basistokens, über ein Dendrogramm. Es werden auch alle aus `id` abgeleitete Tokens („Kinder“) gezeigt und über eine Beispielsicht kann das Token im Kontext des Trainingskorpus betrachtet werden.

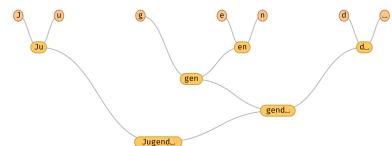


Abb. 27: Tokenstammbaum des Tokens `Jugend`.

7.2.2. Unterseite `/token/embedding-space`

Visualisiert den latenten Embedding-Raum des ausgewählten Modells. Dabei kann sowohl der zweidimensionale (wie in Abb. 22) als auch der dreidimensionale Raum dargestellt und navigiert werden. Die grafische Oberfläche und interaktive Visualisierung wird in deck.gl[68] GPU beschleunigt ausgeführt, sodass auch Animationen zwischen verschiedenen Repräsentationen flüssig betrachtet werden können. Neben der Färbung nach Token-ID wie in bisherigen Abb. kann jeder Punkt auch nach Suffix, Präfix, Byteanzahl, Groß- und Kleinbuchstaben- sowie Ziffernanteil und Wortanzahl eingefärbt werden. Eine Legende beschreibt die Bedeutung der Farben, und ein Histogramm zeigt die Verteilung über alle Tokens auf. Durch Klick im Diagramm können Tokens ausgewählt und hervorgehoben werden, sodass Cluster evident werden.

7.2.3. Unterseite `/chat`

Enthält das Haupt-Inferenzinterface. Auf dieser Seite können das kausale und antikausale Modell genutzt werden, um Texte in respektive gegen die Leserichtung zu vervollständigen. Ebenfalls können die Tokenwahrscheinlichkeiten für ein beliebiges Token angezeigt werden.

8. Fazit und Ausblick

In diesem Projekt ist es mir gelungen, eigene direktionalkomplementäre große Sprachmodelle zu trainieren. Die beiden auf GPT-2 basierenden Modelle `causa11` und `anticausa11` illustrieren bereits jetzt Unterschiede in der richtungsabhängigen Textverarbeitung. Die interaktive Webanwendung leistet zusammen mit den gezeigten Statistiken und der komparativen Visualisierung latenter Räume der beiden Modelle einen Beitrag in Richtung erklärbarer KI-Modelle.

Es konnte gezeigt werden, dass das Training von großen deutschsprachigen antikausalen Sprachmodellen möglich ist und dass ihre Generalisierungsfähigkeit zur Inferenzzeit mit kausalen Sprachmodellen vergleichbar ist. Zugleich konnten elementare Unterschiede in der Repräsentation der Embedding-Schichten, sowohl im Token-Embedding als auch der Positionsencodierung gemessen und statistisch ausgewertet werden.

Die Richtungssensitivität verschiedener Datensätze im Bezug auf das Finetuning konnte ebenfalls untersucht und evaluiert werden. Insgesamt scheinen kausale Modelle für alle angebrachten Anwendungsgebiete bei gleicher Anzahl von Trainingsschritten und unveränderten Parametern einen geringeren Verlust aufzuweisen; der Leistungsunterschied der beiden Modelle unterscheidet sich dagegen stark zwischen Domänen und Datensätzen. Diese Erkenntnis ist kompatibel mit dem intuitiven menschlichen Verständnis von Text und Textinferenz.

Jedoch ist das Projekt nur ein anfänglicher Schritt auf dem Weg zu praktisch einsetzbaren direktionalkomplementären Sprachmodellen und der vollständigen Analyse von Unterschieden und Gemeinsamkeiten kausaler und antikausaler autoregressiver Textinferenz.

Die Welt von sprachverarbeitender KI entwickelt sich rasant, sodass noch während der Projektentwicklung mit Fineweb2[69] ein qualitativ hochwertigerer, da extensiver und feinerer gefilterter, deutschsprachiger Datensatz veröffentlicht wurde, welcher bei Nutzung im Pre-Training die Ausgaben der Modelle insbesondere qualitativ stark verbessern könnte. Ein Training der neuen Modelle `anticausal-fw2` und `causal-fw2` sowie die Aktualisierung der statistischen Evaluationen ist daher bereits geplant.

In Zukunft kann der Fokus dabei darauf liegen, weitere neue Erkenntnisse, auch im Bereich der Computerlinguistik, bezüglich Direktionalität natürlicher Sprache durch Training und Auswertung weiterer Sprachmodelle und Datensätze zu gewinnen. Hierzu wird ein Finetuning der Modelle auf den Projekt Gutenberg-Datensatz fiktionaler Literatur[70] angestrebt. Es gilt auch, die latenten Räume dieser spezialisierten Modelle zu visualisieren sowie durch Eingabe weiterer Tokensequenzen ins Modell auch kontextualisierte Eingaben räumlich einzuordnen.

Geplant wird auch ein Training eines Mamba[71]-Modells auf Grundlage von mamba.py[72] zum Zweck einer Reproduktion von und Untersuchung der Modellunabhängigkeit der getroffenen Aussagen über Direktionalität. Theoretisch geht auch die Suche nach kogenerativer „Zusammenarbeit“ der trainierten Modelle weiter, indem überprüft wird, inwiefern die beispielsweise abwechselnde Tokeninferenz basierend auf einem gegebenen Eingabefragment zu Reduktion der direktionalen Perplexität führen kann. Zudem wird aktiv an der Webanwendung weiterentwickelt, etwa um weiterführende Inferenzmodi auf Grundlage von Constrained Generation[73] oder Beam Search[74] durch eine offene Implementierung allgemein zur Verfügung zu stellen. Unter der Annahme der Gültigkeit neuronaler Skalierungsgesetze[75,76] auf die präsentierten Ergebnisse stellt die antikausale Rückinferenz aber schon jetzt eine vielversprechende neuartige Möglichkeit der KI-Nutzung dar.

9. Danksagung

Ich danke meinem Projektbetreuer, Herrn Prof. Dr. André Grüning, für wertvolle Hinweise zum Training von KI-Modellen und weiterführenden Ausarbeitungsideen und für eine fachliche Überprüfung der vorliegenden Arbeit.

Zudem danke ich meinem Vater, Norman Wojak, für eine sprachliche und gestalterische Überprüfung der Arbeit. Ich danke der Hochschule Stralsund sowie meinem Projektbetreuer für die Bereitstellung des für Training und Inferenz genutzte GPU-Cluster und das zur Projektarbeit benötigte Material. Analog danke ich meiner Vorgesetzten im FJN (Freiwilliges Jahr in Wissenschaft, Technik und Nachhaltigkeit), Silke Krumrey, dafür, an der Hochschule am Projekt arbeiten zu können.

Ich danke auch allen Entwickler:innen der in der Open-Source-Anwendung und für das Modelltraining genutzten freien Softwarebibliotheken, ohne die die Ausarbeitung in ihrer aktuellen Form nicht möglich gewesen wäre. So danke ich auch allen Wissenschaftler:innen, insbesondere den Autor:innen direkter und transitiver Referenzen dieser Arbeit, ohne deren Einsatz die moderne KI-Entwicklung ausgeblieben wäre.

10. Quellen- und Literaturverzeichnis

Sofern nicht anders angegeben, handelt es sich bei allen Abbildungen um eigene.

- [1] A. Vaswani *u. a.*, „Attention is All you Need“, in *Neural Information Processing Systems*, 2017.
- [2] W. X. Zhao *u. a.*, „A Survey of Large Language Models“, 2024, Verfügbar unter: <https://arxiv.org/abs/2303.18223>
- [3] C.-C. Lin, A. Jaech, X. Li, M. R. Gormley, und J. Eisner, „Limitations of Autoregressive Models and Their Alternatives“, 2021, Verfügbar unter: <https://arxiv.org/abs/2010.11939>
- [4] S. Gong *u. a.*, „Scaling Diffusion Language Models via Adaptation from Autoregressive Models“, 2024, Verfügbar unter: <https://arxiv.org/abs/2410.17891>
- [5] Z. ul Abideen, „Autoregressive Models for Natural Language Processing“. Zugegriffen: 12. Februar 2025. Verfügbar unter: <https://medium.com/%40zaiinn440/autoregressive-models-for-natural-language-processing-b95e5f933e1f>
- [6] T. A. Chang und B. K. Bergen, „Language Model Behavior: A Comprehensive Survey“, *Computational Linguistics*, Bd. 50, Nr. 1, S. 293–350, 2024, doi: 10.1162/coli_a_00492.
- [7] S. Yu *u. a.*, „Reverse Modeling in Large Language Models“, 2024, Verfügbar unter: <https://arxiv.org/abs/2410.09817>
- [8] H. Naveed *u. a.*, „A Comprehensive Overview of Large Language Models“, 2024, Verfügbar unter: <https://arxiv.org/abs/2307.06435>
- [9] S. Wolfram, „What is ChatGPT Doing... and Why Does It Work?“, 2023, Zugegriffen: 4. Dezember 2024. Verfügbar unter: <https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/>
- [10] A. Karpathy, „Deep Dive into LLMs like ChatGPT“. Zugegriffen: 8. Februar 2025. Verfügbar unter: <https://www.youtube.com/watch?v=7xTGNNLPyMI>
- [11] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, und I. Sutskever, „Language Models are Unsupervised Multitask Learners“, 2019. Verfügbar unter: <https://api.semanticscholar.org/CorpusID:160025533>
- [12] A. Karpathy, „NanoGPT“, *GitHub repository*, 2022, Zugegriffen: 10. November 2024. Verfügbar unter: <https://github.com/karpathy/nanoGPT>
- [13] X. Liu, H.-F. Yu, I. Dhillon, und C.-J. Hsieh, „Learning to Encode Position for Transformer with Continuous Dynamical Model“, 2020, Verfügbar unter: <https://arxiv.org/abs/2003.09229>
- [14] J. L. Ba, „Layer normalization“, *arXiv preprint arXiv:1607.06450*, 2016.

- [15] R. Xiong u. a., „On layer normalization in the transformer architecture“, in *International Conference on Machine Learning*, 2020, S. 10524–10533.
- [16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, und R. Salakhutdinov, „Dropout: a simple way to prevent neural networks from overfitting“, *The journal of machine learning research*, Bd. 15, Nr. 1, S. 1929–1958, 2014.
- [17] P. Baldi und P. J. Sadowski, „Understanding dropout“, *Advances in neural information processing systems*, Bd. 26, 2013.
- [18] M. Wornow, „Transformer Math (Part 1) - Counting Model Parameters“. Zugegriffen: 13. Februar 2025. Verfügbar unter: <https://michaelwornow.net/2024/01/18/counting-params-in-transformer>
- [19] H. Yao, D.-l. Zhu, B. Jiang, und P. Yu, „Negative log likelihood ratio loss for deep neural network classification“, in *Proceedings of the Future Technologies Conference (FTC) 2019: Volume 1*, 2020, S. 276–282.
- [20] P. Zhou, X. Xie, Z. Lin, und S. Yan, „Towards Understanding Convergence and Generalization of AdamW“, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Bd. 46, Nr. 9, S. 6486–6493, 2024, doi: 10.1109/TPAMI.2024.3382294.
- [21] I. Loshchilov und F. Hutter, „Decoupled Weight Decay Regularization“, 2019, Verfügbar unter: <https://arxiv.org/abs/1711.05101>
- [22] D. P. Kingma, „Adam: A method for stochastic optimization“, *arXiv preprint arXiv:1412.6980*, 2014.
- [23] I. Loshchilov und F. Hutter, „SGDR: Stochastic Gradient Descent with Warm Restarts“, 2017, Verfügbar unter: <https://arxiv.org/abs/1608.03983>
- [24] Y. Bengio, P. Simard, und P. Frasconi, „Learning long-term dependencies with gradient descent is difficult“, *IEEE Transactions on Neural Networks*, Bd. 5, Nr. 2, S. 157–166, 1994, doi: 10.1109/72.279181.
- [25] R. Pascanu, „Understanding the exploding gradient problem“, *arXiv preprint arXiv:1211.5063*, 2012.
- [26] J. Zhang, T. He, S. Sra, und A. Jadbabaie, „Why gradient clipping accelerates training: A theoretical justification for adaptivity“, *arXiv preprint arXiv:1905.11881*, 2019.
- [27] A. Krogh und J. Hertz, „A simple weight decay can improve generalization“, *Advances in neural information processing systems*, Bd. 4, 1991.
- [28] G. Zhang, C. Wang, B. Xu, und R. Grosse, „Three Mechanisms of Weight Decay Regularization“, 2018, Verfügbar unter: <https://arxiv.org/abs/1810.12281>

- [29] J. Chen *u. a.*, „LLMArena: Assessing Capabilities of Large Language Models in Dynamic Multi-Agent Environments“, 2024, Verfügbar unter: <https://arxiv.org/abs/2402.16499>
- [30] J. Abadji, P. Ortiz Suarez, L. Romary, und B. Sagot, „Towards a Cleaner Document-Oriented Multilingual Crawled Corpus“, *arXiv e-prints*, S. arXiv:2201.06642, Jan. 2022.
- [31] Wikimedia Deutschland e. V., „Deutsche Wikipedia, Datenextraktion vom 1. November 2024“. Verfügbar unter: <https://dumps.wikimedia.org/dewiki/>
- [32] D. Shapiro, „Convert Wikipedia dumps into plaintext files“. Verfügbar unter: <https://github.com/daveshap/PlainTextWikipedia>
- [33] Y. Shibata *u. a.*, „Byte pair encoding: A text compression scheme that accelerates pattern matching“, 1999.
- [34] A. Macijauskas, „Tokenizers deep dive“, 2024, Zugegriffen: 20. Februar 2025. Verfügbar unter: <https://augustasmacijauskas.github.io/personal-website/posts/tokenizers-deep-dive/tokenizers-deep-dive.html>
- [35] A. Paszke *u. a.*, „Automatic differentiation in PyTorch“, in *NIPS-W*, 2017.
- [36] L.-K. Schulz, „Cloud Computing for Education: Design and Implementation of a Platform Solution for Dynamic Provisioning of Computing Power and Software with Kubernetes“.
- [37] S. Li *u. a.*, „PyTorch Distributed: Experiences on Accelerating Data Parallel Training“, 2020, Verfügbar unter: <https://arxiv.org/abs/2006.15704>
- [38] F. Chaubard, D. Eddy, und M. J. Kochenderfer, „Beyond Gradient Averaging in Parallel Optimization: Improved Robustness through Gradient Agreement Filtering“, 2024, Verfügbar unter: <https://arxiv.org/abs/2412.18052>
- [39] „Der t-Test“, in *Quantitative Methoden: Einführung in die Statistik*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, S. 43–117. doi: 10.1007/978-3-540-33308-1_3.
- [40] P. Virtanen *u. a.*, „SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python“, *Nature Methods*, Bd. 17, S. 261–272, 2020, doi: 10.1038/s41592-019-0686-2.
- [41] L. Kennedy-Shaffer, „Before p < 0.05 to Beyond p < 0.05: Using History to Contextualize p-Values and Significance Testing“, *The American Statistician*, Bd. 73, Nr. sup1, S. 82–90, 2019, doi: 10.1080/00031305.2018.1537891.
- [42] S. Goodman, „A Dirty Dozen: Twelve P-Value Misconceptions“, *Seminars in Hematology*, Bd. 45, Nr. 3, S. 135–140, 2008, doi: <https://doi.org/10.1053/j.seminhematol.2008.04.003>.
- [43] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*, 2. Aufl. New York: Routledge, 1988. doi: 10.4324/9780203771587.

- [44] Wikipedia, „Wikipedia:Namenskonventionen/Kategorien — Wikipedia, die freie Enzyklopädie“. Zugegriffen: 10. Februar 2025. Verfügbar unter: <https://de.wikipedia.org/wiki/Wikipedia:Namenskonventionen/Kategorien#Allgemeines>
- [45] UpsetMango6411 u. a., „Quick question: can someone confirm whether this is Indonesian?“. Zugegriffen: 10. Februar 2025. Verfügbar unter: <https://www.reddit.com/r/indonesian/comments/l1m8atg/comment/mc4nljh/>
- [46] H. Riza, „Resources report on languages of indonesia“, in *Proceedings of the 6th Workshop on Asian Language Resources*, 2008.
- [47] Deutscher Bundestag, „BundesGit Gesetze Tools“. Zugegriffen: 16. Februar 2025. Verfügbar unter: <https://github.com/bundestag/gesetze>
- [48] Deutscher Bundestag, „Drucksachen der 1. - 19. Wahlperiode“, 2021. Zugegriffen: 16. Februar 2025. Verfügbar unter: <https://www.bundestag.de/services/opendata>
- [49] G. Van Rossum und F. L. Drake Jr, *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.
- [50] Y. Wang, H. Huang, C. Rudin, und Y. Shaposhnik, „Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMap, and PaCMAP for Data Visualization“, *Journal of Machine Learning Research*, Bd. 22, Nr. 201, S. 1–73, 2021, Verfügbar unter: <http://jmlr.org/papers/v22/20-1061.html>
- [51] „Hauptkomponentenanalyse (PCA)“, in *Multivariate Statistik in der Ökologie: Eine Einführung*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, S. 105–123. doi: 10.1007/978-3-540-37706-1_9.
- [52] L. v. d. Maaten und G. Hinton, „Visualizing data using t-SNE“, *Journal of machine learning research*, Bd. 9, Nr. Nov, S. 2579–2605, 2008.
- [53] L. McInnes, J. Healy, und J. Melville, „UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction“, 2020, Verfügbar unter: <https://arxiv.org/abs/1802.03426>
- [54] M. Gruber, „Why you should not rely on t-SNE, UMAP or TriMAP“. Zugegriffen: 8. Februar 2025. Verfügbar unter: <https://towardsdatascience.com/why-you-should-not-rely-on-t-sne-umap-or-trimap-f8f5dc333e59/>
- [55] A. Clark, „Pillow (PIL Fork) Documentation“. Verfügbar unter: <https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf>
- [56] J. D. Hunter, „Matplotlib: A 2D graphics environment“, *Computing in Science & Engineering*, Bd. 9, Nr. 3, S. 90–95, 2007, doi: 10.1109/MCSE.2007.55.

- [57] M. L. Waskom, „seaborn: statistical data visualization“, *Journal of Open Source Software*, Bd. 6, Nr. 60, S. 3021, 2021, doi: 10.21105/joss.03021.
- [58] L. Blume, „DEversAI“. Zugegriffen: 9. Februar 2025. Verfügbar unter: <https://deversai.vercel.app/>
- [59] L. Blume, „Quelltext für das Jugend forscht-Projekt DEversAI“. Zugegriffen: 17. Februar 2025. Verfügbar unter: <https://github.com/leo848/deversai>
- [60] R. Harris, A. Faubert, T. L. Hau, B. McCann, und andere, „Svelte – cybernetically enhanced web apps“. Zugegriffen: 3. Januar 2024. Verfügbar unter: <https://svelte.dev/>
- [61] R. Harris, A. Faubert, T. L. Hau, B. McCann, und andere, „SvelteKit: Web development, streamlined.“. Zugegriffen: 2. Januar 2024. Verfügbar unter: <https://kit.svelte.dev/>
- [62] A. Hejlsberg, „TypeScript: JavaScript with types“. Zugegriffen: 2. Januar 2024. Verfügbar unter: <https://www.typescriptlang.org/>
- [63] G. Bierman, M. Abadi, und M. Torgersen, „Understanding TypeScript“, in *European Conference on Object-Oriented Programming*, 2014, S. 257–281.
- [64] S. Ramírez, „FastAPI“. Verfügbar unter: <https://github.com/fastapi/fastapi>
- [65] J. Pasche, „Uberspace - Hosting auf Asteroids“. Zugegriffen: 17. Februar 2025. Verfügbar unter: <https://uberspace.de/de/>
- [66] I. Fette und A. Melnikov, „The websocket protocol“, 2011.
- [67] M. Bostock, „D3.js - Data-Driven Documents“. Verfügbar unter: <http://d3js.org/>
- [68] Uber, „deck.gl: WebGL2 powered geospatial visualization layers“. Zugegriffen: 30. Dezember 2024. Verfügbar unter: <https://deck.gl/>
- [69] G. Penedo *u. a.*, „FineWeb2: A sparkling update with 1000s of languages“. Verfügbar unter: <https://huggingface.co/datasets/HuggingFaceFW/fineweb-2>
- [70] H. Reuters, „Projekt Gutenberg“. Zugegriffen: 17. Februar 2025. Verfügbar unter: <https://www.projekt-gutenberg.org/>
- [71] A. Gu und T. Dao, „Mamba: Linear-Time Sequence Modeling with Selective State Spaces“, 2024, Verfügbar unter: <https://arxiv.org/abs/2312.00752>
- [72] A. Torres-Leguet, „mamba.py: A simple, hackable and efficient Mamba implementation in pure PyTorch and MLX.“. Verfügbar unter: <https://github.com/alexndrTL/mamba.py>
- [73] D. Banerjee, T. Suresh, S. Ugare, S. Misailovic, und G. Singh, „CRANE: Reasoning with constrained LLM generation“, 2025, Verfügbar unter: <https://arxiv.org/abs/2502.09061>

- [74] S. Wiseman und A. M. Rush, „Sequence-to-Sequence Learning as Beam-Search Optimization“, 2016, Verfügbar unter: <https://arxiv.org/abs/1606.02960>
- [75] J. Kaplan *u. a.*, „Scaling Laws for Neural Language Models“, 2020, Verfügbar unter: <https://arxiv.org/abs/2001.08361>
- [76] Y. Bahri, E. Dyer, J. Kaplan, J. Lee, und U. Sharma, „Explaining neural scaling laws“, *Proceedings of the National Academy of Sciences*, Bd. 121, Nr. 27, S. e2311878121, 2024, doi: 10.1073/pnas.2311878121.