

Hochschule Darmstadt

– Fachbereich Mathematik und
Naturwissenschaften –
– Fachbereich Informatik –

Fake-Rezensionen durch Textgenerierung: Einsatz von LLMs zur Erstellung eines gelabelten Datensatzes für maschinelles Lernen

Abschlussarbeit zur Erlangung des akademischen Grades
Master of Science (M. Sc.)

vorgelegt von

Johanna Bopst

Matrikelnummer: 1119036

Referentin : Prof. Dr. Melanie Siegel

Korreferent : Prof. Dr. Thomas März

Ausgabedatum: : 10.07.2025

Abgabedatum: : 09.01.2026

ERKLÄRUNG

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Während der Vorbereitung dieser Arbeit habe ich DeepL Write und ChatGPT für die Sprachkorrektur verwendet. Nach der Nutzung dieser Tools habe ich den Inhalt überprüft und bearbeitet und übernehme die volle Verantwortung für den Inhalt der Veröffentlichung.

Frankfurt, 4. Januar 2026

Johanna Bopst

ABSTRACT

Online product reviews have a substantial impact on purchasing decisions, which makes manipulative or artificially generated reviews an increasingly relevant problem. The rising availability of powerful LLMs enables the generation of synthetic reviews in high quality and large scale, further complicating the detection of opinion spam. The aim of this thesis is to create a high-quality, labeled dataset of German product reviews consisting of both real and LLM-generated texts, which can serve as a research basis for the development of detection methods.

For this purpose, genuine Amazon product reviews as well as synthetic reviews generated by different LLMs via few-shot prompting and fine-tuning were used. Stratified sampling was applied to ensure that the statistical distributions of both subsets are comparable with respect to categories, rating scales and text length. The quality of the final dataset was assessed both statistically and linguistically. To evaluate the realism of the generated reviews, two classification models (logistic regression and RoBERTa) were trained and compared.

The results show that RoBERTa achieves an accuracy of 90 %, outperforming the linear model in distinguishing between real and generated reviews. The hardest to detect generated reviews originate from the fine-tuned *GPT-4.1 Nano* model. Overall, this thesis demonstrates that the constructed dataset exhibits realistic distributions and is suitable for the evaluation of modern classification models. The resulting dataset closes an existing research gap in the German language domain and provides a basis for further work in automated opinion-spam detection.

Keywords: Opinion Spam, Large Language Models, Few-Shot Prompting, Fine-Tuning, RoBERTa, Logistic Regression, German-Language Dataset

ZUSAMMENFASSUNG

Online-Produktrezensionen haben einen erheblichen Einfluss auf Kaufentscheidungen, wodurch manipulierte oder künstlich erzeugte Bewertungen zu einem wachsenden Problem werden. Die zunehmende Verfügbarkeit leistungsfähiger LLMs ermöglicht es, synthetische Rezensionen in hoher Qualität und großer Menge zu generieren, was die Erkennung von Opinion Spam zusätzlich erschwert. Ziel dieser Arbeit ist die Erstellung eines qualitativ hochwertigen, gelabelten Datensatzes aus deutschsprachigen Produktrezensionen, der sowohl reale als auch durch LLMs generierte Texte umfasst und als Forschungsgrundlage für die Entwicklung von Detektionsverfahren im Bereich Opinion Spam dienen kann.

Hierzu wurden echte Amazon-Produktrezensionen sowie synthetische Rezensionen, die durch verschiedene LLMs mittels Few-Shot-Prompting und Fine-Tuning generiert wurden, verwendet. Mittels stratifizierten Samplings wurde sichergestellt, dass die statistischen Verteilungen beider Teilmengen hinsichtlich Kategorien, Bewertungsskalen und Textlängen vergleichbar sind. Die Qualität des finalen Datensatzes wurde sowohl statistisch als auch im Bezug auf sprachliche Diversität bewertet. Zur Prüfung der Realitätsnähe der generierten Rezensionen wurden zwei Klassifikationsmodelle (logistische Regression und RoBERTa) trainiert und miteinander verglichen.

Die Ergebnisse zeigen, dass RoBERTa mit einer Genauigkeit von 90 % besser zwischen realen und generierten Rezensionen unterscheiden kann als das lineare Modell. Die am schwersten erkennbaren generierten Rezensionen stammen dabei vom feinabgestimmten GPT-4.1 Nano-Modell. Insgesamt wurde deutlich, dass der erstellte Datensatz realistische Verteilungen aufweist und sich zur Evaluierung moderner Klassifikationsmodelle eignet. Die bereitgestellte Datengrundlage schließt damit eine Forschungslücke im deutschsprachigen Raum und bildet die Basis für weiterführende Arbeiten im Bereich der automatisierten Opinion-Spam-Detektion.

Schlüsselwörter: Opinion Spam, Large Language Models, Few-Shot-Prompting, Fine-Tuning, RoBERTa, logistische Regression, deutschsprachiger Datensatz

INHALTSVERZEICHNIS

I	Thesis	
1	Einleitung	2
1.1	Motivation	2
1.2	Ziel der Arbeit	2
1.3	Struktur	3
2	Theroretische Grundlagen	4
2.1	Opinion Spam	4
2.2	Large Language Models	6
2.2.1	Transformer-Modellarchitektur	6
2.2.1.1	Encoder	7
2.2.1.2	Decoder	8
2.2.1.3	Attention	9
2.2.2	Typen	11
2.2.3	Verwendete Transformer-Modelle zur Textgenerierung	11
2.2.3.1	Chat-GPT	12
2.2.3.2	LLaMA	13
2.2.3.3	Gemma	14
2.2.4	Modellsteuerung	14
2.2.4.1	Fine-Tuning	14
2.2.4.2	Prompt-Engineering	16
2.2.4.3	Decoding Strategie	17
2.3	Klassifikationsverfahren und Evaluationsmetriken	18
2.3.1	Logistische Regression	18
2.3.2	RoBERTa	21
2.3.3	Performance-Metriken	22
2.3.4	Self-BLEU	23
3	Related Work	25
4	Datensatzerstellung	27
4.1	Experimentelle Umgebung	27
4.2	Pipeline	27
4.3	Datenvorverarbeitung	28
4.4	Rezensionsgenerierung	30
4.4.1	Few-Shot-Prompting	30
4.4.2	Fine-Tuning-basierte Generierung	32
5	Datensatzanalyse und Klassifikation	36
5.1	Qualitätsanalyse des Datensatzes	36
5.1.1	Statistische Analyse	36
5.1.2	Sprachliche Vielfalt	37
5.2	Klassifikation der Rezensionen	38
5.2.1	Modelltraining	39
5.2.2	Ergebnisse und Evaluation	41

6	Fazit und Ausblick	48
II Appendix		
A	Anhang	51
A.1	Beispiel-Prompt	51
A.2	Rezension für Fine-Tuning	53
A.3	Originalverlauf ohne Glättung des Fine-Tunings	53
A.4	Beispielrezensionen pro Modell	56
A.5	Generierte Rezensionen der Kategorien Books, Mobile Apps und Music	57
A.6	False Negatives	57
A.7	Rezensionlänge je Sternebewertung	58
	Literatur	59

ABBILDUNGSVERZEICHNIS

Abbildung 2.1	Transformer-Architektur bestehend aus <i>Encoder</i> und <i>Decoder</i> nach Vaswani et al. [64].	7
Abbildung 2.2	<i>Attention</i> -Mechanismus. Quelle: [64]	10
Abbildung 2.3	Nur A und B werden trainiert. Quelle: [22]	15
Abbildung 2.4	Visuelle Darstellung des Prompt-Engineering-Prozesses: <i>LLMs</i> , die auf umfangreichen Daten trainiert wurden, nutzen Anweisungen und Kontexte als zentrale Elemente für die Gestaltung von Prompts. Quelle: [52]	16
Abbildung 2.5	Vergleich zwischen Zero-, One- und Few-Shot sowie dem traditionellen Fine-Tuning. Quelle: [3]	17
Abbildung 2.6	Pre-Training und Fine-Tuning von <i>BERT</i> . Quelle: [9]	22
Abbildung 4.1	Prozess der Datensatzerstellung.	27
Abbildung 4.2	Verteilung der Rezensionslängen.	29
Abbildung 4.3	Loss und Accuracy Fine-Tuning <i>Music</i>	34
Abbildung 5.1	Verteilung der Modelle im Datensatz.	37
Abbildung 5.2	Verteilungen der echten und generierten Rezensionen im Bezug auf Rezensionslänge und Sternebewertung.	37
Abbildung 5.3	F1-Validation pro Epoche.	40
Abbildung 5.4	Train und Validation Loss pro Epoche.	40
Abbildung 5.5	Konfusionsmatritzen.	42
Abbildung 5.6	<i>ROC</i> -Kurven.	43
Abbildung 5.7	Venn-Diagramme.	45
Abbildung 5.8	Accuracy je Wortanzahl-Bin für beide Klassifikatoren: Die Prozentwerte geben den Anteil der Rezensionen für jeden Bin der Test-Daten an.	46
Abbildung 5.9	Accuracy je Sternebewertung für beide Klassifikatoren: Die Prozentwerte geben den Anteil der Rezensionen für jedes Rating der Test-Daten an.	47
Abbildung A.1	Loss und Accuracy Fine-Tuning <i>Books</i>	53
Abbildung A.2	Loss und Accuracy Fine-Tuning <i>Mobile Apps</i>	54
Abbildung A.3	Loss und Accuracy Fine-Tuning <i>Music</i>	54
Abbildung A.4	Loss und Accuracy Fine-Tuning <i>PC</i>	54
Abbildung A.5	Loss und Accuracy Fine-Tuning <i>Toys</i>	55
Abbildung A.6	durchschnittliche Rezensionslänge je Sternebewertung.	58

TABELLENVERZEICHNIS

Tabelle 2.1	Konfusionsmatrix für binäre Klassifikation.	23
Tabelle 2.2	Kennzahlen zur binären Klassifizierung unter Verwendung der Notation von Tabelle 2.1.	23
Tabelle 4.1	Übersicht der verwendeten Features.	30
Tabelle 4.2	Fine-Tuning-Evaluation.	35
Tabelle 5.1	BLEU- und n-Gramm-Präzisionen der generierten und echten Rezensionen.	38
Tabelle 5.2	Top 10 Wörter nach Koeffizienten für logistische Regression.	39
Tabelle 5.3	Performance-Metriken für beide Klassifikatoren.	41
Tabelle 5.4	Kategorieweise Accuracy für RoBERTa und logistische Regression.	42
Tabelle 5.5	Anzahl der Fehlklassifikationen und Accuracy nach Modell für logistische Regression und RoBERTa.	44
Tabelle A.1	3-Sterne-Beispielrezensionen der Kategorie <i>Music</i> aus dem Datensatz.	56
Tabelle A.2	Generierte Rezensionen der Kategorie <i>Books</i> , <i>Mobile Apps</i> und <i>Music</i> aus dem Datensatz.	57
Tabelle A.3	Anzahl der <i>False Negatives</i> pro Modell und Produktkategorie.	57

LISTINGS

4.1	Beispiel für Prompt mit zwei Few-Shot-Beispielen.	31
4.2	Beispiel Rezension im JSON-Format für Fine-Tuning.	33
A.1	vollständiger Few-Shot-Prompt für die Rezensionsgenierung der Kategorie <i>Toys</i>	51
A.2	Rezension im JSON-Format für Fine-Tuning.	53

ABKÜRZUNGSVERZEICHNIS

AUC	Area Under the Curve
BERT	Bidirectional Encoder Representations from Transformers
BLEU	Bilingual Evaluation Understudy
FFN	Feed-Forward Network
GeGLU	Gaussian Error Gated Linear Unit
GPT	Generative Pre-Training Transformer
GWGD	Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen
LLaMA	Large Language Model Meta AI
LoRA	Low-Rank Adaption
LLMs	Large Language Models
MLM	Masked Language Modeling
NSP	Next Sentence Prediction
NLP	Natural Language Processing
PEFT	Parameter-Efficient Fine-Tuning
ReLU	Rectified Linear Unit
RNNs	rekurrenten neuronalen Netzen
RoBERTa	Robustly Optimized BERT Pretraining Approach
ROC	Receiver Operating Characteristic
SGD	Stochastic Gradient Descent
SwiGLU	Swish Gated Linear Unit
TF-IDF	Term Frequency-Inverse Document Frequency

Teil I

THESIS

EINLEITUNG

Dieses Kapitel führt in das Thema ein, erläutert die Motivation der Arbeit, formuliert die zugrunde liegenden Forschungsfragen und beschreibt den strukturellen Aufbau der Thesis.

1.1 MOTIVATION

Online-Produktrezensionen haben einen unmittelbaren Einfluss auf Kaufentscheidungen und sind für viele Nutzer*innen ein zentraler Bestandteil der Informationssuche vor dem Kauf [57]. Gleichzeitig nimmt das Vorkommen manipulierter Rezensionen („Opinion Spam“) seit Jahren zu. Dieser Trend wird durch den breiten Einsatz von Large Language Models (LLMs) verstärkt, da automatisiert generierte Rezensionen inzwischen in großen Mengen, schnell und kostengünstig erstellt werden können. Neben dem durch manipulierte Rezensionen entstehenden Vertrauensverlust [10] haben falsche Bewertungen auch wirtschaftliche Auswirkungen. Sie beeinflussen beispielsweise Ranking-Algorithmen auf Plattformen und können dadurch die Sichtbarkeit und den Absatz von Produkten verändern [14].

Damit wächst die Relevanz verlässlicher Methoden gegen Opinion Spam. Allerdings existieren bislang kaum geeignete Datensätze, die zur Entwicklung und Evaluierung solcher Klassifikationsmodelle genutzt werden können. Verfügbare Referenzdatensätze sind häufig stark limitiert, etwa durch eine zu geringe Anzahl an Rezensionen [44]. Zudem basieren bestehende Datensätze meistens auf menschlich verfasstem Opinion Spam, während maschinell generierter Spam bislang kaum systematisch erfasst wurde. Insbesondere im deutschsprachigen Raum existiert aktuell kein Datensatz, der generierte Rezensionen realistisch abbildet und somit als Forschungsgrundlage dienen kann.

Die Bereitstellung eines solchen Datensatzes schafft folglich eine Grundlage, von der Studierende, Forschende und Entwickler*innen profitieren können, um Modelle zu entwickeln und zu optimieren, die Opinion Spam in Zukunft wirksamer adressieren.

1.2 ZIEL DER ARBEIT

Ziel dieser Arbeit ist die Erstellung eines qualitativ hochwertigen, gelabelten Datensatzes aus deutschen Produktrezensionen, der sowohl echte als auch künstlich generierte Texte umfasst und als Grundlage für Forschung im Bereich automatisierter Opinion-Spam-Detektion nutzbar ist.

Die zentrale Leitfrage lautet:

Wie lässt sich ein qualitativ hochwertiger, gelabelter Datensatz aus echten und LLM-generierten Rezensionen erstellen?

Daraus ergeben sich folgende untergeordnete Forschungsfragen:

- Welche Rolle spielen Prompt-Design und Modellwahl bei der Generierung künstlicher, realistischer Rezensionen?
- Wie kann eine realitätsnahe Datenverteilung in Bezug auf Produktkategorien, Bewertungsskalen und Textlänge sichergestellt werden?
- Welche spezifischen Herausforderungen entstehen bei der Generierung und Klassifikation deutschsprachiger Rezensionen?
- Wie lässt sich die Qualität des Datensatzes validieren?

Im Zentrum steht dabei nicht die Optimierung einzelner Generierungs- oder Klassifikationsmodelle, sondern die systematische Untersuchung und Evaluation, wie ein solcher Datensatz konstruiert werden muss, um realitätsnah, auswertbar und wissenschaftlich belastbar zu sein.

1.3 STRUKTUR

Kapitel 2 stellt die theoretischen Grundlagen bereit. Dazu werden der Forschungsgegenstand Opinion Spam und die technischen Grundlagen moderner Sprachmodelle erläutert, bevor anschließend auf die im Rahmen der Arbeit konkret verwendeten LLMs eingegangen wird. Darüber hinaus werden die für diese Arbeit relevanten Klassifikationsmodelle sowie Evaluationsmetriken vorgestellt.

Weiterführend ordnet Kapitel 3 die Arbeit in den Stand der Forschung ein. Es werden existierende Ansätze, Datensätze und Verfahren zur Erkennung von Opinion Spam betrachtet und deren Limitationen herausgearbeitet.

In Kapitel 4 wird die Erstellung des Datensatzes beschrieben. Dazu wird die vollständige Pipeline, bestehend aus Datenvorverarbeitung, Prompt-basierter Textgenerierung und Fine-Tuning, detailliert dargestellt. Dieses Kapitel bildet den zentralen methodischen Beitrag der Arbeit.

Kapitel 5 widmet sich der Analyse und Bewertung des erstellten Datensatzes. Zunächst erfolgt eine statistische und sprachliche Qualitätsanalyse. Anschließend werden zwei unterschiedliche Klassifikationsmodelle trainiert und evaluiert, um die Realitätsnähe der generierten Rezensionen empirisch zu überprüfen.

Das letzte Kapitel 6 fasst die wichtigsten Erkenntnisse zusammen, diskutiert Limitationen und zeigt Potenziale für zukünftige Forschung auf.

Das folgende Kapitel legt das theoretische Fundament dieser Arbeit und schafft damit die konzeptionelle Grundlage für die Datensatzerstellung und -analyse. Zunächst wird mit dem Themenfeld Opinion Spam der Forschungsgegenstand eingeordnet und es werden zentrale Begriffe sowie Mechanismen manipulativer Rezensionen erläutert. Darauf aufbauend werden die Funktionsweise und architektonischen Prinzipien von LLMs beschrieben, einschließlich der in dieser Arbeit eingesetzten Modelle und ihrer Steuerungsmöglichkeiten durch Fine-Tuning und Prompt-Engineering. Im Anschluss werden die eingesetzten Klassifikationsverfahren und Evaluationsmetriken vorgestellt, die dazu dienen, die Qualität des Datensatzes zu bewerten.

2.1 OPINION SPAM

Begriffsdefinition

Nach Liu [29] bezeichnet der Begriff „Opinion Spam“ gefälschte oder irreführende Meinungen, auch bekannt als Fake-Rezensionen, die darauf abzielen, Leser oder automatische Systeme in die Irre zu führen. Dies kann beispielsweise dadurch geschehen, dass bestimmte Produkte positiv bewertet werden, um sie zu bewerben (*Hype Spam*), oder dass andere Produkte negativ bewertet werden, um deren Ruf zu schädigen (*Defaming Spam*). Bei der Erkennung von Opinion Spam geht es darum, herauszufinden, ob eine Bewertung manipuliert und damit irreführend ist. Dabei lassen sich drei Arten unterscheiden:

spam reviews: Bei der Erkennung von *Spam Reviews* geht es darum, zu analysieren, ob eine Bewertung von einer autorisierten Person verfasst wurde.

spammer detection: Dies bezieht sich auf die Erkennung individueller Spammer. Es werden Personen detektiert, die Fake-Rezensionen verfassen.

group spam: Hierbei handelt es sich um eine Gruppe von Personen, die in das Schreiben von Fake-Rezensionen involviert ist [47].

Der Fokus dieser Arbeit liegt ausschließlich auf der Generierung von *Spam Reviews*, wobei primär die Erstellung der eigentlichen Rezensionstexte untersucht wird. Auf die Einbeziehung von Metadaten wurde bewusst verzichtet, da diese den Fokus von der inhaltlichen Textanalyse hin zur *Spammer* oder *Group Spam Detection* verschieben würden. Der erstellte Datensatz konzentriert sich daher ausschließlich auf Rezensionstexte aus verschiedenen

Kategorien und bildet damit eine geeignete Grundlage für die Analyse text-basierter Spam-Erkennung.

Bedeutung

Laut Untersuchungen aus dem Jahr 2015 waren schätzungsweise bis zu 30 % der Online-Bewertungen gefälscht [37]. Mit dem rapiden technologischen Fortschritt hat auch die Generierung künstlich erstellter Inhalte im Internet für Konsumenten deutlich zugenommen. Da Online-Rezensionen eine signifikante Rolle im Hinblick auf die Kaufentscheidung von Nutzer*innen einnehmen, wächst die Anzahl gefälschter Rezensionen [56]. Laut einer Studie des Pew Research Centers [57] geben 82% der US-amerikanischen Bevölkerung an, zumindest gelegentlich Rezensionen vor dem Erwerb von Produkten oder Dienstleistungen zu lesen. Rund 40 % der Befragten geben an, dies ausnahmslos zu tun. Die Erzeugung von Rezensionen kann auf menschlicher oder maschineller Ebene erfolgen. Während menschengemachte Fälschungen in der Regel zu einem kurzfristigen Anstieg der Zugriffsraten, der Verkaufszahlen und der Position in den Suchergebnissen führen, ist langfristig ein Abfall zu beobachten [19]. Dieser kann durch maschinelle und technologische Entwicklungen in signifikanter Weise kompensiert werden. Die Erzeugung automatisierter Rezensionen verhindert einen solchen Abfall, da durchgängig kostengünstig Rezensionen generiert werden können.

Fake-Rezensionen können sich auf unterschiedlichen Ebenen nachteilig auswirken. In diesem Zusammenhang sind insbesondere drei Aspekte hervorzuheben:

1. VERTRAUENSVERLUST UND UNZUFRIEDENHEIT DER KONSUMENTEN:
Die Zunahme gefälschter Bewertungen hat eine Destabilisierung des Vertrauens in Online-Rezensionen zur Folge. Dies hat einen negativen Einfluss auf die Kaufabsicht der Konsument*innen [10]. Gefälschte Bewertungen können außerdem dazu führen, dass Kund*innen falsche Erwartungen an ein Produkt entwickeln. Die daraus resultierende Diskrepanz zwischen dem erwarteten und dem tatsächlich wahrgenommenen Wert des Produkts beeinträchtigt die Kund*innenzufriedenheit. Diese ergibt sich maßgeblich aus dem Vergleich zwischen wahrgenommenem und erwartetem Wert [69].
2. MANIPULATION VON RANKINGS DURCH FAKE REVIEWS:
Bewertungen werden häufig zur Ranking-Berechnung von Produkten herangezogen [14]. Eine große Anzahl positiver oder negativer Rezensionen kann das Ranking manipulieren. Dies kann dazu dienen, eigene Produkte künstlich aufzuwerten oder Konkurrenzprodukte abzuwerten und damit weniger sichtbar zu machen.
3. WIRTSCHAFTLICHER EINFLUSS:
Fake-Bewertungen können auch erhebliche wirtschaftliche Folgen für Unternehmen haben. Eine Studie von Luca [32] zeigt, dass der Anstieg der Sternebewertung bei Yelp um eine Einheit den Gewinn eines Restaurants um 5 % bis 9 % erhöhen kann. Somit beeinträchtigen

gefälschte Bewertungen sowohl die Entscheidungsfindung der Konsument*innen als auch den wirtschaftlichen Erfolg der Unternehmen.

Rechtliche Maßnahmen

Aufgrund der hohen Relevanz reagierte die Europäische Union auf die Problematik und implementierte verschiedene Regulierungen. Gemäß der am 28. Mai 2022 in Kraft getretenen Modernisierungsrichtlinie (Omnibus-Richtlinie – (EU) 2019/2161) [51] ist es untersagt, Online-Bewertungen zu erwerben, zu veräußern oder anderweitig zu manipulieren. Des Weiteren sind Unternehmen dazu verpflichtet, die Handhabung von Bewertungen sowie die Durchführung einer entsprechenden Echtheitsprüfung transparent zu gestalten und deutlich sichtbar zu kennzeichnen. Im Falle grenzüberschreitender Verstöße werden Geldbußen von mindestens 4 % des Jahresumsatzes der Unternehmen verhängt. Gemäß dem Digital Services Act [65] sind Anbieter sehr großer Online-Plattformen seit 2024 dazu verpflichtet, Risikoprüfungen durchzuführen.

2.2 LARGE LANGUAGE MODELS

LLMs sind große Sprachmodelle, die zur Kategorie der *Foundation Models* (Basismodelle) gehören. Sie werden auf riesigen Textkorpora trainiert, um Strukturen, Muster und semantische Zusammenhänge zu lernen. Sie können verschiedenste Aufgaben in der Sprachverarbeitung ausführen, darunter Textgenerierung, Zusammenfassungen und Übersetzungen. Ein großer Vorteil der *Foundation Models* ist ihre Zero-Shot-Funktionalität. Das heißt, sie können Aufgaben bewältigen, ohne speziell dafür trainiert worden zu sein. Die 2017 von Vaswani et al. [64] eingeführten Transformer stellten einen Wendepunkt in der Entwicklung neuronaler Sprachmodelle dar. Im Vergleich zu den zuvor genutzten rekurrenten neuronalen Netzen (RNNs) erlaubt die Architektur eine vollständige Parallelisierung während des Trainings sowie einen besseren Umgang mit Langzeitabhängigkeiten [64]. Die Transformer-Architektur bildet die Grundlage für die Sprachmodelle, die im Rahmen dieser Arbeit zur Generierung von Fake-Rezensionen verwendet werden.

2.2.1 *Transformer-Modellarchitektur*

Die parallele Verarbeitung der Eingabedaten macht die im Folgenden beschriebene Architektur möglich. In seiner ursprünglichen Form besteht das Transformer-Modell aus jeweils sechs identischen *Encoder* und *Decoder Layers* (Kodierer- und Dekodierer-Schichten) und wurde lediglich für Übersetzungsaufgaben entwickelt. Wie in Abbildung 2.1 rechts und links zu sehen, bestehen beide aus *Self Attention Stacks* (gestapelten Selbst-Aufmerksamkeits-Schichten) und *Pointwise Fully Connected Layers* (punktweise vollständig verbundene Schichten).

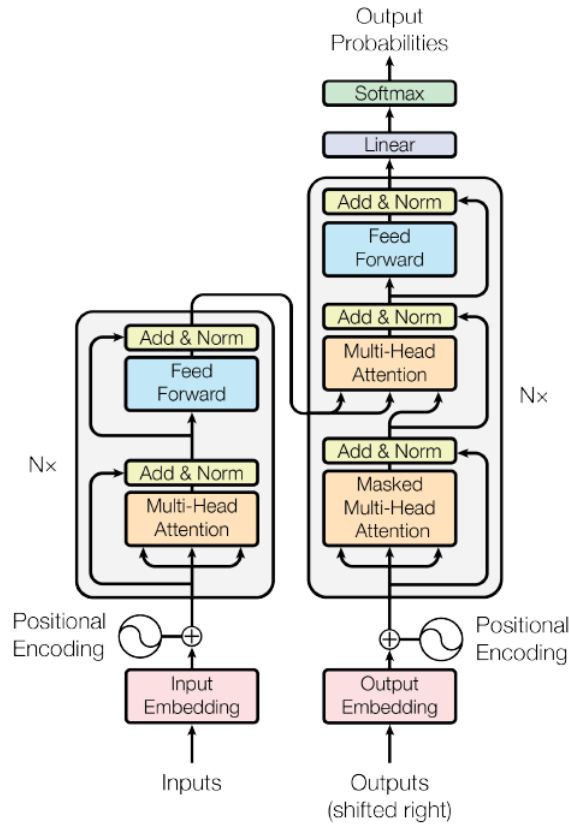


Abbildung 2.1: Transformer-Architektur bestehend aus *Encoder* und *Decoder* nach Vaswani et al. [64].

2.2.1.1 Encoder

Im ersten Schritt werden Eingaben (z. B. Wortsequenzen) tokenisiert und anschließend in der *Embedding*-Schicht (Einbettungsschicht) in numerische Vektoren (*Embeddings*) transformiert. Die *Embedding*-Schicht produziert eine 512-dimensionale Vektorrepräsentation für jedes Wort in der Sequenz. Um Informationen über die Positionen der Wörter in der Sequenz beizubehalten, werden den *Input Embeddings* am unteren Ende der *Encoder*- und *Decoder Layer* Positionskodierungen hinzugefügt. Die Positionskodierungen besitzen die gleiche Dimension d_{model} wie die *Embeddings*, sodass sie direkt addiert werden können. Vaswani et al. [64] verwenden dafür in ihrer Arbeit folgende Sinus- und Kosinusfunktionen:

$$\text{PE}_{(pos,2i)} = \sin\left(pos/10000^{2i/d_{\text{model}}}\right) \quad (2.1)$$

$$\text{PE}_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d_{\text{model}}}\right) \quad (2.2)$$

Dabei bezeichnet pos die Position des Tokens im Satz und i die Dimension. Jede Dimension der Positionskodierung entspricht somit einer Sinus- oder Kosinusfunktion mit spezifischer Frequenz. Diese Kodierungen ermöglichen es dem Modell, die Reihenfolge der Tokens im Input zu berücksichtigen.

Der vorverarbeitete Input wird an den *Encoder* weitergeleitet, welcher aus einem Stapel von $N = 6$ identischen Schichten besteht. Die Aufgabe dieser Schichten besteht darin, den Eingabedaten eine höherdimensionale Repräsentation zuzuordnen, die die im Modell erlernten semantischen und strukturellen Informationen des Inputs abbildet. Eine einzelne Schicht besteht aus zwei zentralen Komponenten, einer *Multi-Head Self-Attention*-Schicht (mehrköpfiger Selbstaufmerksamkeitsmechanismus) und einem vollständig verbundenen *Feed-Forward-Network* (vorwärtsgerichtetes Netz), die beide normalisiert werden, bevor ihre Ausgaben an die nächste Schicht weitergegeben werden. Zusätzlich wird eine Residual-Verbindung implementiert. Diese verbessern die Optimierung Neuronaler Netze, in dem der Input einer Schicht zum Output der nachfolgenden Schichten addiert wird. Mittels *Shortcut Connections* (Sprungverbindungen) können dabei eine oder mehrere Schichten übersprungen werden [18]. Der Output des *Encoders* wird an die *Multi-Head-Attention*-Schicht des *Decoders* weitergeleitet.

2.2.1.2 Decoder

Der *Decoder* besteht, analog zum *Encoder*, aus einem Stapel von $N = 6$ identischen Schichten. Neben den beiden auch im *Encoder* verwendeten Teilschichten enthält der *Decoder* eine dritte Teilschicht, welche eine *Multi-Head-Attention* mit dem Output des *Encoder*-Stapels ausführt auch (*Cross-Attention*).

Wie im *Encoder* werden um jede Teilschicht Residualverbindungen gelegt, gefolgt von einer Normalisierungsschicht. Die *Self-Attention*-Teilschicht des *Decoders* wird zusätzlich so angepasst, dass einzelne Positionen nicht auf zukünftige Positionen zugreifen können. Diese Maskierung (*Masked Self-Attention*) sowie die Verschiebung der Ausgabesequenz um eine Position stellen sicher, dass die Vorhersage für eine Position i ausschließlich von den bereits bekannten Ausgaben der vorherigen Positionen abhängt [64]. Auf diese Weise wird die autoregressive Generierung der Sequenz ermöglicht. Die Ausgabe des *Decoders* wird an eine lineare Schicht weitergeleitet, die in der Funktion eines Klassifikators agiert. In der Folge wird sie an eine Softmax-Funktion übergeben, die Wahrscheinlichkeiten für jedes Wort berechnet. Die Gewichte werden anschließend gemäß der herkömmlichen Methode mittels Backpropagation aktualisiert, nachdem der Verlust quantifiziert wurde. Ein signifikanter Vorteil von Transformern besteht darin, dass *Encoder* und *Decoder* getrennt und als unabhängige Modelle verwendet werden können (vgl. Kapitel 2.2.2).

2.2.1.3 Attention

Der zentrale Bestandteil der Transformer-Architektur ist der *Attention*-Mechanismus. Dieser adressiert die beiden wesentlichen Einschränkungen früherer Ansätze wie *RNNs*. Durch die Möglichkeit der parallelen Verarbeitung von Sequenzen überwinden Transformer die Notwendigkeit einer schrittweisen Verarbeitung und ermöglichen zugleich die Erfassung langfristiger Abhängigkeiten innerhalb von Textsequenzen [64].

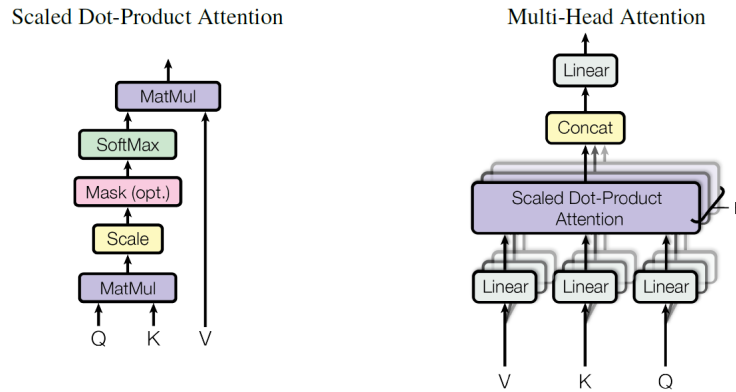
Das Grundprinzip des *Attention*-Mechanismus besteht darin, dass das Modell seine Aufmerksamkeit dynamisch auf die jeweils relevanten Teile der Eingabesequenz richtet. Dadurch kann es die Beziehungen zwischen Tokens gezielt gewichten und kontextabhängige Informationen effektiver verarbeiten. Der Mechanismus setzt sich aus den drei Komponenten zusammen. Es werden eine Abfrage Q , ein Schlüssel K und ein Wert V benötigt. Eine Funktion kann als Zuordnung einer Abfrage und einer Reihe von Schlüssel-Wert-Paaren zu einem Output beschrieben werden, wobei Abfrage, Schlüssel, Werte und Output Vektoren sind. Der Output wird als gewichtete Summe der Werte berechnet, wobei das jeweilige Gewicht durch eine Kompatibilitätsfunktion bestimmt wird, die die Ähnlichkeit zwischen der Abfrage und dem zugehörigen Schlüssel bewertet. Der in *Encodern* verwendete *Self-Attention*-Mechanismus ermöglicht es einem Modell, jeden einzelnen Token in einer Eingabesequenz mit den anderen Tokens dieser Sequenz zu assoziieren. Es wird die Ähnlichkeit eines Tokens zu allen anderen Tokens einschließlich zu sich selbst innerhalb einer Sequenz berechnet. Dabei stammen alle Vektoren aus dem *Embedding*-Input oder der vorherigen Schicht. Im Gegensatz dazu werden beim *Attention*-Mechanismus im *Decoder* nur die Abfragen aus der vorherigen Schicht des *Decoders* abgeleitet, während die Schlüssel und Werte aus dem Output des *Encoders* stammen. Im Folgenden soll der Ablauf eines solchen Mechanismus Schritt für Schritt erläutert werden.

Scaled Dot-Product Attention

Abbildung 2.2 zeigt die *Attention*-Mechanismen innerhalb der Transformer-Architektur. Der Input der *Scaled Dot-Product Attention* (skalierte Punktprodukt-Aufmerksamkeit) besteht aus Abfragen und Schlüsseln der Dimension d_k und Werten der Dimension d_v . Die *Attention*-Funktion wird gleichzeitig für eine Menge von Abfragen berechnet, die gemeinsam in einer Matrix Q zusammengefasst werden. Ebenso werden die Schlüssel K und Werte V in entsprechenden Matrizen gebündelt.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V \quad (2.3)$$

Formel (2.3) zeigt die *Attention*-Berechnung. Die Funktion berechnet das Skalarprodukt zwischen der Abfragematrix und Transponierten der Schlüsselmatrix. Dieser Schritt misst die Ähnlichkeit zwischen jeder Abfrage und jedem Schlüssel. Um zu vermeiden, dass die Softmax-Funktion nicht in Re-

Abbildung 2.2: *Attention*-Mechanismus. Quelle: [64]

gionen abrutscht, in denen die Gradienten extrem klein werden, wird das Skalarprodukt mit $(\frac{1}{\sqrt{d_k}})$ skaliert. Die Softmax-Funktion transformiert die berechneten Scores in Wahrscheinlichkeiten und erzeugt dadurch eine Verteilung, die angibt, wie stark die Abfrage sich auf die einzelnen Schlüssel fokussieren soll. Sie drückt aus, wie stark jedes Eingabewort zur Repräsentation eines Zielwortes beiträgt. Diese gewichtete Verteilung wird anschließend auf die Werte-Matrix V angewendet, um die finale Ausgabe der *Attention* zu berechnen. Dadurch entsteht eine Ausgabe, bei der die Werte, die stärker mit den relevanten Schlüsseln korrespondieren, ein höheres Gewicht erhalten, wodurch effektiv eine kontextbezogene Darstellung für jedes Token erstellt wird.

Multi-Head-Attention

Anstatt eine einzige *Attention*-Berechnung auf die d_{model} -dimensionalen Abfrage-, Schlüssel- und Wertevektoren durchzuführen, verwendet der Transformer mehrere parallele *Attention*-Berechnungen, sogenannte *Heads* (Köpfe) (vgl. Abb. 2.2 rechts). Dazu werden die Abfragen, Schlüssel und Werte jeweils h -mal mit unterschiedlichen, gelernten linearen Gewichtsmatrizen projiziert. Jeder *Head* kann dadurch unterschiedliche Aspekte der Eingabe erfassen, beispielsweise syntaktische im einen und semantische Beziehungen im anderen. Die Ausgaben der h *Attention-Heads* werden anschließend wie folgt zusammengeführt:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (2.4)$$

Dabei hat jeder head_i eine eigene *Attention*-Berechnung auf einer individuellen Projektion der Eingaben:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (2.5)$$

Mit den Parameter-Matrizen folgender Größen: $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, $W^O \in \mathbb{R}^{h \cdot d_v \times d_{\text{model}}}$.

Dabei W^O ist die Gewichtsmatrix der letzten linearen Schicht nach der Konkatenation aller *Attention-Heads* und dient somit zur Gewichtung der kombinierten Ausgabe.

Vaswani et al. [64] verwenden $h = 8$ parallele *Heads* und wählen $d_k = d_v = d_{\text{model}}/h = 64$.

Position-wise Feed-Forward Network

Neben dem *Attention*-Mechanismus hat jede *Encoder*- und *Decoder*-Schicht ein Feed-Forward Network (FFN), das für jede Position separat und identisch angewendet wird. Die linearen Transformationen sind zwar an verschiedenen Positionen gleich, verwenden jedoch von Schicht zu Schicht unterschiedliche Parameter. Das FFN besteht aus zwei linearen Transformationen mit einer dazwischenliegenden Rectified Linear Unit (ReLU)-Aktivierung [64].

2.2.2 Typen

Die Architektur von Transformermodellen kann je nach Anwendungszweck unterschiedlich ausgestaltet sein. Grundsätzlich lassen sich drei Haupttypen unterscheiden, die sich in ihrer Struktur und den damit verbundenen Einsatzgebieten unterscheiden:

encoder-only: Sie basieren ausschließlich den *Encoder* des Transformers und werden primär für Aufgaben des Textverständnisses eingesetzt, wie etwa Textklassifizierung oder Informationsextraktion. Beispiele für solche Modelle sind BERT [9] oder RoBERTa [30].

decoder-only: Sie verwenden ausschließlich den *Decoder* und werden für Textgenerierung verwendet. Beispiele sind GPT [48], LLaMA [63] oder Gemma [61].

encoder-decoder: Sie nutzen *Decoder* und *Encoder* und sind beispielsweise für Übersetzungsaufgaben konzipiert. Ein populäres Beispiel ist T5 (Text-to-Text Transfer Transformer) [49].

2.2.3 Verwendete Transformer-Modelle zur Textgenerierung

Die im vorherigen Kapitel erläuterte Transformer-Architektur bildet die Grundlage zahlreicher LLMs, die für unterschiedlichste Aufgabenbereiche konzipiert wurden. Im Folgenden werden die Modelle vorgestellt, die für die Generierung der Rezensionen eingesetzt wurden. Dabei wird insbesondere auf architekturenspezifische Anpassungen und Besonderheiten eingegangen, die diese Modelle auszeichnen und für den Einsatz im Kontext der Textsynthese relevant machen. Alle eingesetzten Modelle basieren dabei auf einer *Decoder-Only*-Architektur.

2.2.3.1 Chat-GPT

Die Einführung des Generative Pre-Training Transformer (GPT) durch Radford et al. [48] stellt einen wesentlichen Fortschritt im Bereich des Natural Language Processing (NLP) dar, insbesondere im Hinblick auf generative Textanwendungen. Im Gegensatz zu dem von Google entwickelten Sprachmodell Bidirectional Encoder Representations from Transformers (BERT) [9], das ausschließlich auf einer *Encoder*-Architektur basiert, verwendet GPT eine rein *Decoder*-basierte Architektur und ist somit generativ ausgelegt. Im Vergleich zur ursprünglichen Transformer-Architektur werden beim GPT 12 *Self-Attention Heads* sowie 768-dimensionale Vektorrepräsentationen implementiert. Zudem verwendet GPT ausschließlich *Masked Self-Attention*, sodass sichergestellt werden kann, dass bei der Generierung eines Tokens ausschließlich auf vorherige Tokens zugegriffen wird. Dieser unidirektionale Informationsfluss unterscheidet GPT grundlegend von der bidirektionalen Architektur des ursprünglichen Transformers [48].

Das Modell wird in einem zweistufigen Verfahren trainiert. Zuerst erfolgt ein unüberwachtes Pre-Training auf einem großen, nicht gelabeltem Textkorpus. In diesem Schritt werden die anfänglichen Parameter des neuronalen Netzwerkmodells erlernt. Anschließend wird ein überwachtes Fine-Tuning mit gelabelten Daten durchgeführt, wobei das Modell auf spezifische Aufgaben wie die Textklassifikation oder das Question Answering angepasst wird. Da die zugrunde liegende Architektur während des Fine-Tunings unverändert bleibt, ist GPT task-agnostisch. Demnach kann es flexibel für verschiedene Aufgaben eingesetzt werden, ohne dass strukturelle Änderungen erforderlich sind [48]. Mittlerweile ist die Trainingsdatenmenge der GPT-Modelle so umfangreich geworden, dass ohne Fine-Tuning und mittels Zero-, One- und Few-Shot-Prompting eine starke Performance erreicht wird [3].

Im Rahmen dieser Arbeit wird das Modell GPT-4.1 Nano zur automatisierten Generierung von Rezensionstexten verwendet. Die GPT-4.1-Modellreihe stellt eine Weiterentwicklung der vorhergehenden Serien dar und integriert mehrere signifikante Verbesserungen. Die umfassende Erweiterung des Kontextfensters stellt die wichtigste Neuerung dar. Während frühere GPT-Modelle maximal 128.000 Token verarbeiten konnten, unterstützt GPT-4.1 erstmals Kontextlängen von bis zu einer Million Token. Dies ermöglicht eine deutlich umfassendere Verarbeitung längerer Dokumente oder zusammenhängender Kontexte [42, 43]. Da OpenAI keine detaillierten Informationen über die Zusammensetzung der Trainingsdaten veröffentlicht, ist unklar, welcher Anteil der Daten nicht englischsprachig ist. GPT-4.1 Nano ist die schnellste und günstigste Version der vierten Generation. Trotz seiner reduzierten Modellgröße bietet es eine hinreichende Leistungsfähigkeit für domänenspezifische Textgenerierungsaufgaben und erfüllt die funktionalen Anforderungen dieser Arbeit in vollem Umfang.

2.2.3.2 LLaMA

Ein weiterer Teil der Rezensionstexte wurde mithilfe zweier Modelle der Reihe Large Language Model Meta AI (LLaMA) generiert. Mit der Veröffentlichung von LLaMA im Jahr 2023 stellte Meta AI eine Familie von LLMs vor, deren Parameteranzahl zwischen sieben und 65 Milliarden liegt [63]. Innerhalb eines LLMs repräsentieren die Parameter die trainierbaren Gewichtungen der Modellarchitektur, beispielsweise in den *Attention*-Mechanismen. Im Gegensatz zu den GPT-Modellen von OpenAI wurde LLaMA ausschließlich auf öffentlich verfügbaren Daten trainiert.

Touvron et al. [63] adaptieren in ihrer Arbeit das Transformer-Design von Vaswani et al. [64] als architektonisches Fundament, implementieren jedoch gezielt eine Reihe architekturbezogener Verbesserungen, die aus späteren Arbeiten hervorgegangen sind. Die folgenden drei Änderungen werden als Hauptanpassungen beschrieben:

pre-normalization: Wie GPT ab der dritten Generation verwendet LLaMA das sogenannte *Pre-Normalization*-Verfahren. Dabei wird im Gegensatz zur ursprünglichen Transformer-Architektur nicht der Output einer Teilschicht, sondern bereits deren Input normalisiert. Es wird die RMSNorm-Funktion von Zhang und Sennrich [70] verwendet.

swiglu aktivierungsfunktion: Die ReLU-Aktivierungsfunktion wird durch die effizientere Swish Gated Linear Unit (SwiGLU)-Funktion ausgetauscht.

rotary embeddings: Statt absoluten *Positional Embeddings* (Positionseinbettungen) werden in jeder Schicht des Netzwerks *Rotary Positional Embeddings* (rotierende Positionseinbettungen) hinzugefügt. Anstelle der Addition eines Positionsvektors wird die Positionsinformation durch eine Rotation des Wortvektors im Vektorraum kodiert [60].

Weitere architektonische Änderungen betreffen eine optimierte und ressourcenschonendere Berechnung der *Attention*. Darüber hinaus zeichnen sich die LLaMA-Modelle durch eine zum Teil deutlich geringere Anzahl an Parametern im Vergleich zu GPT-Modellen aus [63]. Die aktuellste LLaMA-3-Serie umfasst Modelle mit einer Größenordnung von acht bis 405 Milliarden Parametern [15].

In dieser Arbeit werden Modelle der dritten Generation (LLaMA 3) eingesetzt.

Bei diesen erfolgte das Training auf einem multilingualen Korpus mit einem Umfang von insgesamt 15 Billionen Tokens. Davon entfallen etwa 5 % auf nicht-englischsprachige Inhalte, die sich auf rund 30 verschiedene Sprachen verteilen. Das Kontextfenster der Modelle beträgt bis zu 128.000 Tokens [15, 35].

Beide verwendeten Modelle *Llama-3.1-SauerkrautLM-70B-Instruct* und *Llama-3.3-70B-Instruct* sind sogenannte Instruct-Versionen. Diese wurden mittels umfangreicher Datensätze aus Anweisungen und entsprechenden

Antworten gezielt für Aufgaben wie Dialogführung und Fragebeantwortung feinjustiert. Dabei orientiert sich das Fine-Tuning an der Methodik von Chung et al. [6]. Beim Modell *Llama-3.1-SauerkrautLM-70b-Instruct* wurden 15% der Modell-Layer durch VAGO Solutions mittels Spectrum Fine-Tuning [16] weiter optimiert, um die multilingualen Fähigkeiten zu verbessern. Hierfür kam ein deutsch-englischer Datensatz zum Einsatz [59].

2.2.3.3 Gemma

Das letzte für die Generierung der Rezensionen eingesetzte LLM ist das *Gemma-3-27b-It* von Google, welches auf 14 Billionen Tokens trainiert wurde. Ähnlich wie das LLaMA-Modell basiert auch dieses grundlegende Transformermodell auf der in Kapitel 2.2.1 vorgestellten Architektur, wurde jedoch an mehreren Stellen modifiziert. Neben vergleichbaren Anpassungen hinsichtlich der *Pre-Normalization* und der Verwendung von *Rotary Embeddings* wie bei den LLaMA-Modellen, ersetzt das Gemma-Modell die ReLU-Aktivierungsfunktion durch eine approximierte Gaussian Error Gated Linear Unit (GeGLU)-Aktivierung [61].

Darüber hinaus wurde bei der hier verwendeten dritten Generation des Gemma-Modells das Kontextfenster auf 128.000 Token erweitert. Zusätzlich kommt eine Kombination aus einer lokalen *Sliding Window Self-Attention* [1] und globaler *Self-Attention* [33] zum Einsatz [62].

Bei der lokalen *Sliding Window Self-Attention* beachtet ein Token nur eine begrenzte Anzahl an umliegenden Tokens innerhalb eines festen Fensters. Das im Rahmen der Arbeit verwendete Modell *Gemma-3-27b-It* ist ein Instruct-Modell.

2.2.4 Modellsteuerung

Im folgenden Abschnitt werden die zentralen Ansätze vorgestellt, die gezielt das Verhalten und die Ausgabe der LLMs zur Generierung realistischer Rezensionen beeinflussen. In dieser Arbeit kommen das Fine-Tuning und das Prompt-Engineering zum Einsatz. Beide Ansätze setzen an unterschiedlichen Ebenen des Modells an. Während beim Fine-Tuning die internen Modellparameter angepasst werden, erfolgt die Steuerung beim Prompt-Engineering ausschließlich über die Gestaltung der Eingabeaufforderung. Abschließend wird die gewählte Decoding-Strategie erläutert, welche durch probabilistische Parameter bestimmt, wie das Modell bei der Textgenerierung das nächste Token auswählt und somit die Art der Ausgabe maßgeblich beeinflusst.

2.2.4.1 Fine-Tuning

Die Grundidee des Fine-Tunings besteht darin, ein bestehendes *Foundation Model* mit einem kleineren, spezifischen Datensatz so nachzutrainieren, dass ein tieferes Verständnis der entsprechenden Domäne erlangt wird. Fine-Tuning stellt sicher, dass die *Foundation Models* ihr allgemeines Wissen beibe-

halten, gleichzeitig jedoch für spezialisierte, domänenspezifische Aufgaben angepasst werden können [68].

Im Rahmen dieser Arbeit wurde das Modell *GPT-4.1 nano* finegetuned. Zwar gibt OpenAI nicht preis, welche Fine-Tuning-Methode verwendet wird, doch lässt sich aufgrund der auf der Website vorgeschlagenen Anzahl an Trainingsbeispielen sowie der Dauer des Fine-Tuning-Prozesses vermuten, dass die *LoRA*-Technik zum Einsatz kommt [41]. Diese wird daher im Folgenden näher erläutert.

LoRA

Low-Rank Adaption (*LoRA*) gehört zu den Parameter-Efficient Fine-Tuning (*PEFT*)-Techniken, bei denen nicht das gesamte Modell, sondern nur eine geringe Anzahl zusätzlicher Parameter optimiert wird, während die übrigen Modellgewichte eingefroren bleiben (vgl. Abb. 2.3). Dabei werden trainierbare Rangzerlegungsmatrizen in jede Schicht der Transformer-Architektur eingefügt, wodurch die Anzahl der zu optimierenden Parameter für nachgelagerte Aufgaben reduziert wird [22].

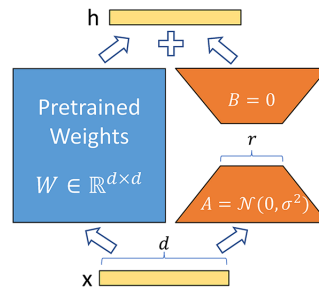


Abbildung 2.3: Nur A und B werden trainiert. Quelle: [22]

Die *LoRA*-Technik nutzt die linearen Abhängigkeiten in den Gewichtsmatrizen aus. Anstatt die gesamte Matrix der Gewichtsänderungen ΔW zu lernen, wird diese in zwei kleinere Matrizen zerlegt:

$$\Delta W = B \cdot A \quad (2.6)$$

mit

- $A \in \mathbb{R}^{r \times k}$,
- $B \in \mathbb{R}^{d \times r}$,
- $r \ll \min(d, k)$

wobei r der Rang-Hyperparameter ist, der deutlich kleiner gewählt wird als die Dimensionen d oder k . Die aktualisierten Gewichte lassen sich dann wie folgt darstellen:

$$W = W_0 + B \cdot A \cdot \frac{\alpha}{r} \quad (2.7)$$

Hierbei ist W_0 die ursprüngliche Gewichtsmatrix des vortrainierten Modells und α ein Skalierungsfaktor zur Feinabstimmung der Gewichtsanpassungen. Für jede domänenspezifische Aufgabe können mittels **LoRA** ein unterschiedliche Adapter erstellt werden. Im vorliegenden Anwendungsfall wird für jede Kategorie ein eigener Adapter eingesetzt.

2.2.4.2 Prompt-Engineering

Als Prompt Engineering wird der Prozess bezeichnet, bei dem Input Queries (sogenannte Prompts) entworfen und verfeinert werden, ohne die Modellparameter anzupassen, um gewünschte Antworten eines **LLMs** zu erzeugen. Die Qualität der Prompts ist dabei maßgeblich für die Effektivität des Modells. Für die Erstellung geeigneter Prompts ist es wichtig, das Ziel klar zu definieren, die Fähigkeiten des Modells einzuschätzen, ein passendes Prompt-Format zu wählen sowie relevanten Kontext bereitzustellen [34]. Prompt-Engineering ist im Vergleich zum Fine-Tuning eine weniger komplexe Methode, **LLMs** domänenspezifisch anzupassen, da das Modell nicht verändert werden muss.

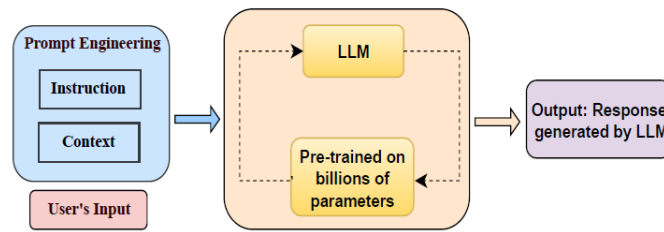


Abbildung 2.4: Visuelle Darstellung des Prompt-Engineering-Prozesses: **LLMs**, die auf umfangreichen Daten trainiert wurden, nutzen Anweisungen und Kontexte als zentrale Elemente für die Gestaltung von Prompts. Quelle: [52]

Von den über 40 verschiedenen Prompting-Techniken [52] wird im Rahmen dieser Arbeit das sogenannte Few-Shot-Prompting verwendet.

Few Shot Prompting

Beim Few-Shot-Prompting werden den Modellen einige wenige Eingabe-Ausgabe-Beispiele bereitgestellt. Dadurch wird ein Verständnis für eine bestimmte Aufgabe vermittelt. Im Gegensatz dazu werden beim Zero-Shot-Prompting keine Beispiele bereitgestellt [3, 52]. Selbst wenige hochwertige Beispiele können die Ergebnisse verbessern. Darüber hinaus können die Auswahl und Zusammensetzung der Prompt-Beispiele das Modellverhalten erheblich beeinflussen. Verzerrungen, wie die Bevorzugung häufiger Wörter, können die Few-Shot-Ergebnisse beeinträchtigen, weswegen ein sorgfältiges Prompt-Engineering entscheidend ist. Wie in Abbildung 2.5 dargestellt, besteht eine Instanz aus einem Kontext und einer gewünschten Vervollständigung (z. B. einem englischen Satz und dessen französischer Übersetzung).

Beim Few-Shot-Ansatz werden dem Modell zunächst K Beispiele mit Kontext und Vervollständigung präsentiert, gefolgt von einem weiteren Beispiel, bei dem nur der Kontext gegeben ist und zu dem das Modell die passende Vervollständigung generieren soll. Die Hauptvorteile von Few-Shot sind eine erhebliche Verringerung des Bedarfs an aufgabenspezifischen Daten und ein geringeres Risiko, das Modell durch Fine-Tuning zu overfitten [3].

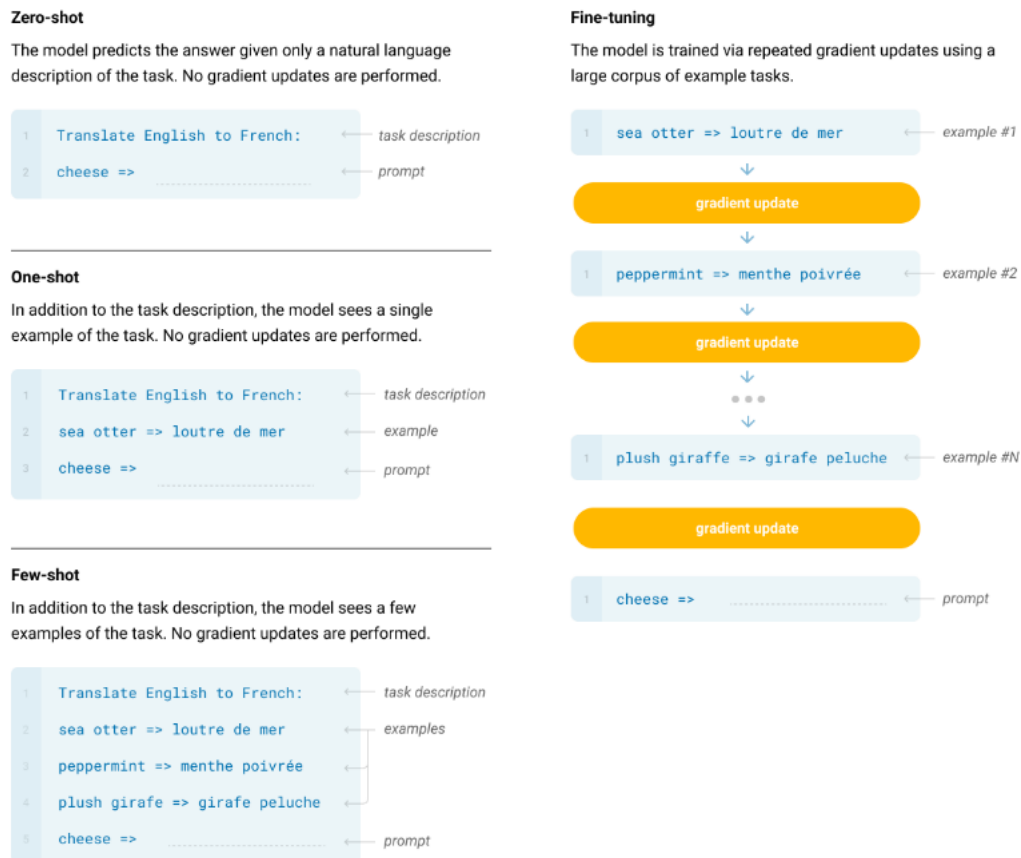


Abbildung 2.5: Vergleich zwischen Zero-, One- und Few-Shot sowie dem traditionellen Fine-Tuning. Quelle: [3]

2.2.4.3 Decoding Strategie

Als Decoding-Strategie wird im Kontext von LLMs die Methode bezeichnet, mit der basierend auf den berechneten Wahrscheinlichkeiten des LLMs die Auswahl des nächsten Tokens (z. B. Wort) erfolgt. Im Rahmen der Textgenerierung berechnet das Modell eine Wahrscheinlichkeitsverteilung über alle potenziell folgenden Tokens, die angibt, mit welcher Wahrscheinlichkeit jedes Token als nächstes generiert wird. Decoding-Strategien legen fest, nach welchen Regeln diese Token ausgewählt werden.

Top-p-Sampling

Im Anwendungsfall dieser Arbeit wird die statistische Decoding-Strategie Top-p-Sampling (auch Nucleus Sampling) verwendet. Sie wurde 2019 von Holtzman et al. [21] entwickelt, um Defizite deterministischer Algorithmen wie Beam Search zu adressieren. Dazu zählen insbesondere häufige Wiederholungen oder zusammenhanglose Kontexte.

Bei sogenannten Sampling-Strategien wird das nächste Wort entsprechend der vom LLM extrahierten Wahrscheinlichkeitsverteilung ausgewählt. Im Gegensatz zum Top-k-Sampling [12] wird beim Top-p-Sampling keine feste Anzahl der wahrscheinlichsten Tokens gewählt, sondern die kleinste Menge an Tokens, deren kumulative Wahrscheinlichkeit den Schwellenwert p überschreitet. Auf diese Weise kann die Anzahl der Tokens in der Auswahl dynamisch an die Wahrscheinlichkeitsverteilung des nächsten Tokens angepasst werden. Je höher p gewählt wird, desto mehr Token werden berücksichtigt. Dies führt dazu, dass die Texte kreativer werden. Holtzman et al. [21] führen aus, dass Werte zwischen 0,9 und 1 am häufigsten gewählt werden.

Trotz dieser statistischen Ansätze bestehen weiterhin Probleme. Die Art und Weise, wie Modelle während des Trainings Wahrscheinlichkeiten für die nächsten Token optimieren, kann dazu führen, dass sie Sequenzen mit Wiederholungen und häufig vorkommenden Wörtern überbewerten, was im Gegensatz zur Verteilung menschlicher Trainingsdaten steht [67].

2.3 KLASSIFIKATIONSVERFAHREN UND EVALUATIONSMETRIKEN

In diesem Kapitel werden die für diese Arbeit relevanten Klassifikationsverfahren sowie die eingesetzten Methoden zur Bewertung ihrer Leistungsfähigkeit beschrieben. Zunächst werden mit der logistischen Regression und **RoBERTa** zwei unterschiedliche Modelle vorgestellt, die exemplarisch für traditionelle statistische Verfahren und moderne Transformer-Architekturen stehen. Anschließend werden die zentralen Evaluationsmetriken erläutert, die zur Beurteilung der Modelle eingesetzt werden. Es folgt eine Erklärung weiterführender Ansätze wie **Self-BLEU**, die zusätzliche Einblicke in die Qualität der Modellvorhersagen und die Diversität generierter Texte liefern.

2.3.1 *Logistische Regression*

Die logistische Regression stellt ein klassisches Verfahren des überwachten maschinellen Lernens dar und zählt zu den diskriminativen Klassifikationsverfahren. In der vorliegenden Arbeit wird sie zur Identifikation von gefälschten Rezensionen innerhalb des erstellten Datensatzes herangezogen. Das Ziel der logistischen Regression besteht darin, die Wahrscheinlichkeit des Eintretens eines bestimmten binären Ereignisses (beispielsweise „echt“ oder „fake“) in Abhängigkeit von einer oder mehreren unabhängigen Variablen zu modellieren. Die Methode wird in Kombination mit **TF-IDF** ver-

wendet, um ein robustes Modell zur Erkennung von Opinion Spam zu entwickeln.

Logistische Funktion

Die logistische Regression ist ein mathematisches Modell, das die Wahrscheinlichkeit $P(y = 1 \mid \mathbf{x})$ beschreibt, dass ein Beobachtungsvektor \mathbf{x} zur Klasse $y = 1$ gehört. Die Basis der hier dargestellten Wahrscheinlichkeitsfunktion ist die Sigmoidfunktion $\sigma(z)$ (auch logistische Funktion), mittels derer die genannte Wahrscheinlichkeit auf dem Intervall $[0, 1]$ abgebildet wird [11, 24].

$$P(y = 1 \mid \mathbf{x}) = \sigma(z) = \frac{1}{1 + e^{-z}}, \quad \text{mit } z = \mathbf{w} \cdot \mathbf{x} + b \quad (2.8)$$

Hierbei bezeichnen:

- $\mathbf{x} \in \mathbb{R}^n$: Merkmalsvektor,
- $\mathbf{w} \in \mathbb{R}^n$: Gewichtungsvektor,
- $b \in \mathbb{R}$: Bias-Term.

Die Entscheidung, ob eine Eingabe zur Klasse $y = 1$ gehört oder nicht, erfolgt anhand einer festen Grenze, der sogenannten Entscheidungsschwelle (*decision boundary*). Der Standardwert dieser Schwelle liegt bei 0,5 [24].

$$\text{decision}(\mathbf{x}) = \begin{cases} 1, & \text{wenn } P(y = 1 \mid \mathbf{x}) > 0,5 \\ 0, & \text{sonst} \end{cases} \quad (2.9)$$

Lernen und Kreuzentropie-Verlustfunktion

Die Optimierung des Modells erfolgt durch Stochastic Gradient Descent (SGD) auf einem annotierten Trainingsdatensatz. Das Ziel besteht darin, die Kreuzentropie-Verlustfunktion zu minimieren, die wie folgt definiert ist:

$$\mathcal{L}(y, \hat{y}) = -[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})] \quad (2.10)$$

Dabei ist:

- $y \in \{0, 1\}$: tatsächliche Klasse (1 für „echt“, 0 für „fake“),
- $\hat{y} = \sigma(\mathbf{w} \cdot \mathbf{x} + b) \in (0, 1)$: die vom Modell vorhergesagte Wahrscheinlichkeit.

Diese Funktion misst die Diskrepanz zwischen der Vorhersage \hat{y} und dem echten Label y . Bei hoher Übereinstimmung zwischen Vorhersage und tatsächlicher Klasse resultieren niedrige Funktionswerte, während starke Diskrepanzen zu hohen Werten führen [24].

Regularisierung

In der Regel erfolgt eine Regulierung des Modells mit dem Ziel der Stabilisierung und der Vermeidung von Overfitting. Die im Rahmen dieser Arbeit verwendete Funktion, die die logistische Regression durchführt, verwendet die L2-Regularisierung (auch Ridge Regression) [72]. Dabei wird eine Strafe für die Regressionskoeffizienten eingeführt, die proportional zu ihren Quadraten ist. Die Summe der Quadrate der Koeffizienten wird an die Verlustfunktion angehängt [17].

TF-IDF

Texte wie Rezensionen müssen vor der Klassifikation in numerische Vektoren überführt werden. Term Frequency-Inverse Document Frequency (TF-IDF) ist ein weit verbreitetes Verfahren in diesem Bereich [54].

Gegeben sei ein Korpus mit einer Menge von Begriffen $T = \{t_1, \dots, t_n\}$, die im Gesamtkorpus vorkommen. Ein Dokument d_i wird durch einen n -dimensionalen Vektor darstellen:

$$\mathbf{x}_i = (x_{i1}, \dots, x_{in})$$

Hierbei beschreibt jede Komponente x_{ij} das Gewicht des Begriffs t_j innerhalb des Dokuments d_i .

Dieses Gewicht setzt sich aus drei Bestandteilen zusammen:

1. **Term Frequency (TF):** Die Häufigkeit des Begriffs t_j im Dokument d_i .
2. **Inverse Document Frequency (IDF):** Eine Maßzahl für die Seltenheit eines Begriffs im Korpus:

$$IDF_j = \log \left(\frac{N}{DF_j} \right)$$

wobei:

- N : Gesamtzahl der Dokumente im Korpus
- DF_j : Anzahl der Dokumente, in denen t_j vorkommt.

3. **Normierung:** Um eine Vergleichbarkeit zwischen Dokumenten unterschiedlicher Länge zu gewährleisten, werden die TF-IDF-Werte auf eine euklidische Länge von 1 normiert. Zusammengesetzt lautet die Formel dann:

$$x_{ij} = TF_{ij} \cdot IDF_j \cdot \left(\sum_j (TF_{ij} \cdot IDF_j)^2 \right)^{-1/2}$$

[54].

Durch diese Gewichtung werden häufig vorkommende, aber wenig informative Wörter abgeschwächt, während seltene, aber bedeutungstragende Begriffe stärker gewichtet werden. Dadurch eignet sich **TF-IDF** besonders gut, um Texte in aussagekräftige numerische Repräsentationen für Klassifikationsmodelle zu überführen.

2.3.2 *RoBERTa*

Robustly Optimized BERT Pretraining Approach (**RoBERTa**) ist eine optimierte Version des von Google im Jahr 2018 vorgestellten Sprachmodells **BERT** [9, 30]. Die **BERT**-Modelle basieren auf der Transformer-Architektur nach Vaswani et al. [64], unterscheiden sich jedoch deutlich von den in dieser Arbeit ebenfalls betrachteten *Decoder-only*-Modellen (Vgl. Kapitel 2.2.3), die primär für Textgenerierung entwickelt wurden. Im Gegensatz dazu handelt es sich bei den **BERT**-Modellen um eine *Encoder-only*-Architektur, die sich besonders für Klassifikationsaufgaben eignet.

Das zentrale Merkmal der **BERT**-Modelle ist sein bidirektionales Sprachverständnis. Das Modell berücksichtigt beim Verarbeiten eines Tokens sowohl den linken als auch den rechten Kontext. Dies unterscheidet das Modell im Hinblick auf die unidirektionale Verarbeitung autoregressiver Modelle wie **GPT**, **LLaMA** oder **Gemma**.

Pre-Training

RoBERTa verfolgt eine optimierte Pre-Training-Strategie. Das Modell wird mittels eines dynamischen Masked Language Modeling (**MLM**) trainiert. Dabei werden 15 % der Tokens in einer Eingabesequenz maskiert indem sie durch ein spezielles [MASK]-Token ersetzt werden. Anstatt einer statischen Festlegung im Vorfeld wird das Maskierungsmuster bei jedem Trainingsdurchlauf dynamisch neu generiert [30].

Ein Beispiel wäre der Satz „The man [MASK] to the store.“, bei dem das Modell das maskierte Wort „went“ rekonstruieren soll. Im Pre-Training wird in 80 % der Fälle das ausgewählte Token durch [MASK] ersetzt, in 10 % der Fälle durch ein zufälliges anderes Token. Bei den verbleibenden 10 % bleibt das Token unverändert.

Diese Strategie verhindert die Diskrepanz zwischen Pre-training und Fine-Tuning, die entsteht, wenn ein [MASK]-Token im Fine-Tuning nicht auftaucht [9].

Im Vergleich zu **BERT** wurden bei **RoBERTa** neben der dynamischen Maskierung mehrere wesentliche Änderungen vorgenommen, um die Leistungsfähigkeit des Modells zu steigern. **RoBERTa** wurde über einen längeren Zeitraum mit mehr Daten und größeren Batches trainiert. Zudem wurde Next Sentence Prediction (**NSP**) entfernt, da es keine nachweisbaren Verbesserungen einbrachte. Des Weiteren wurde **RoBERTa** auf längeren Sequenzen trainiert, wodurch das Modell in der Lage ist, weitreichendere Abhängigkeiten innerhalb von Texten zu erfassen [30].

Fine-Tuning

Das Fine-Tuning erfolgt, indem aufgabenspezifische Eingaben und Ausgaben in das Modell integriert und sämtliche Parameter End-to-End optimiert werden. Die im Pre-Training verwendeten Eingabeformate „Satz A“ und „Satz B“ lassen sich dabei direkt auf unterschiedliche Aufgaben übertragen. Neben Satzpaaren bei Paraphrasierungsaufgaben, Hypothese-Prämisse-Paaren bei Inferenzaufgaben oder Frage-Text-Paaren bei der Beantwortung von Fragen genügt bei Klassifikationsaufgaben ein einzelner Text, der mit einem leeren zweiten Segment kombiniert wird. Auf der Ausgabeseite werden Token-Repräsentationen für tokenbasierte Aufgaben wie Fragebeantwortung verwendet. Für Klassifikationsaufgaben wird hingegen das Embedding des [CLS]-Tokens als globale Repräsentation der gesamten Sequenz genutzt und in eine einfache Klassifikationsschicht eingespeist [9].

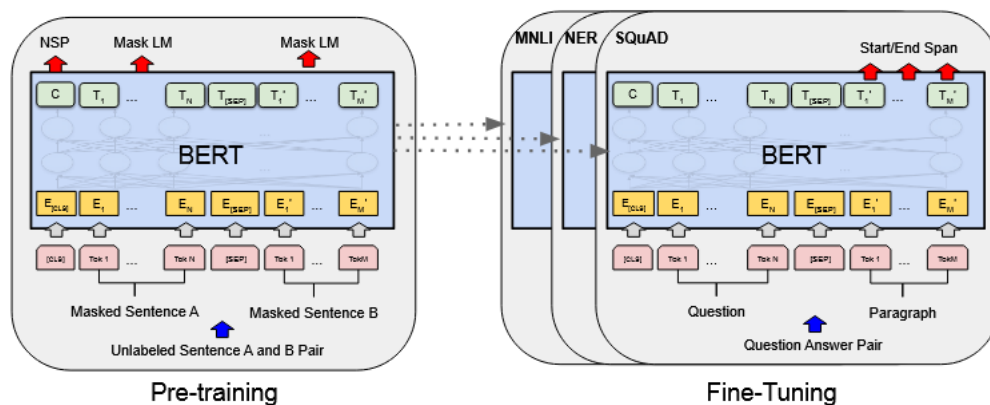


Abbildung 2.6: Pre-Training und Fine-Tuning von BERT. Quelle: [9]

Bild 2.6 verdeutlicht die Prozesse. Bis auf die Output-Layer ist die Architektur beim Pre-Training und Fine-Tuning gleich. Die gleichen im Pre-Training gelernten Modellparameter werden zur Initialisierung der Modelle für verschiedene Downstream-Aufgaben genutzt. Während des Fine-Tunings werden alle Parameter weiter angepasst. [CLS] ist ein spezieller Token, der jedem Eingabebeispiel vorangestellt wird. [SEP] ist ein Trennzeichen [9].

2.3.3 Performance-Metriken

Im vorangegangenen Abschnitt wurde die Funktionsweise der genutzten Klassifikatoren erörtert. Um ihre Performance evaluieren zu können, werden im Folgenden relevante Performance-Maße eingeführt. Die sogenannte Konfusionsmatrix in Tabelle 2.1 beschreibt das Ergebnis einer Klassifizierungsaufgabe vollständig. Jede Zeile der Matrix steht für die Instanzen einer tatsächlichen Klasse und jede Spalte für die Instanzen einer vorhergesagten Klasse. *FN* (*False Negative*) bezeichnet beispielsweise positive Ergebnisse, die fälschlicherweise als negative Ergebnisse vorhergesagt wurden [5]. Im Rah-

men dieser Arbeit sind dies echte Rezensionen, die als "fake" vorhergesagt werden.

		Predicted Value		Total
		Positive	Negative	
True Value	Positive	TP	FN	$TP + FN$
	Negative	FP	TN	$FP + TN$
Total		$TP + FP$	$FN + TN$	N

Tabelle 2.1: Konfusionsmatrix für binäre Klassifikation.

Wichtige Performance-Kennzahlen für die binäre Klassifikation können aus der Konfusionsmatrix abgeleitet werden. Tabelle 2.2 zeigt alle relevanten Maße und ihren Evaluationsfokus auf [5, 58].

Maß	Formel	Evaluationsfokus
Accuracy	$\frac{TP+TN}{TP+FN+FP+TN}$	Gesamteffektivität des Klassifikators
Precision	$\frac{TP}{TP+FP}$	Verhältnis zwischen tatsächlichen und vorhergesagten positiven Labels
Recall (Sensitivity)	$\frac{TP}{TP+FN}$	Fähigkeit des Klassifikators, positive Labels korrekt zu identifizieren
F1-Score	$\frac{2 \cdot TP}{2 \cdot TP + FP + FN}$	harmonisches Mittel aus Precision und Recall
AUC	$\frac{1}{2} \left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP} \right)$	Fähigkeit des Klassifikators, Fehlklassifikationen zu vermeiden

Tabelle 2.2: Kennzahlen zur binären Klassifizierung unter Verwendung der Notation von Tabelle 2.1.

2.3.4 Self-BLEU

Self-Bilingual Evaluation Understudy (BLEU) ist eine Metrik zur Erfassung und Quantifizierung der Textdiversität, die 2018 von Zhu et al. [71] vorgestellt wurde. Sie ist eine Abwandlung des BLEU-Scores [45], der für die Messung der Übersetzungsqualität verwendet wird. Anders als beim traditionellen BLEU, bei dem der n-Gramm-Overlap zwischen einer Menge generierter Übersetzungen und einer Menge von Referenztexten (z. B. menschlichen Übersetzungen) gemessen wird, nutzt Self-BLEU die vom Modell erzeugten Ausgaben selbst als Referenzen. Dadurch wird die Ähnlichkeit innerhalb der generierten Texte bewertet. Ein Modell, das sehr ähnliche oder nahezu identische Texte produziert, weist folglich einen hohen Self-BLEU-Wert auf.

Die BLEU-Metrik ist definiert als

$$BLEU-n = BP \cdot \exp \left(\sum_{i=1}^n \lambda_i \log p_n \right), \quad (2.11)$$

wobei p_n die Präzision für n-Gramme der Ordnung i darstellt und λ_i Gewichtungsfaktoren sind. p_n ist ein Genauigkeitsmaß, das den Anteil der n -Gramme in der Kandidatenmenge bestimmt, die auch in der Referenzmenge vorkommen.

Die Brevity Penalty BP wird definiert als:

$$BP = \min \left(1, \frac{\text{Output-Länge}}{\text{Referenz-Länge}} \right). \quad (2.12)$$

Das Auslassen von Wörtern wird durch die Kürzungstrafe bestraft. Ist die Ausgabe zu kurz, reduziert BP den BLEU-Score [27]. BLEU ist damit das geometrische Mittel der n-Gramm-Präzisionen p_n , multipliziert mit der Kürzungstrafe [28].

Üblicherweise wird die maximale n-Gramm-Ordnung auf $n = 4$ gesetzt. Die daraus resultierende Metrik wird als BLEU-4 bezeichnet.

Der Self-BLEU berechnet sich analog, nur dass als Referenzmenge nicht menschliche Referenzen, sondern die anderen generierten Texte dienen.

Für jeden einzelnen generierten Text wird der BLEU-Score mit allen übrigen Texten als Referenz berechnet [27, 28].

RELATED WORK

Dieses Kapitel bietet einen Überblick zum aktuellen Forschungsstand. Es werden relevante Datensätze und Arbeiten zu Opinion Spam vorgestellt.

Datensätze

Für die Erkennung von Opinion Spam existieren zwar einige Datensätze, diese weisen jedoch alle spezifische Einschränkungen auf. Für die deutsche Sprache steht bislang kein entsprechender Datensatz zur Verfügung. Der vermutlich am häufigsten genutzte Datensatz stammt von Ott et al. [44]. Er umfasst 400 echte Rezensionen zu den 20 beliebtesten Hotels einer US-amerikanischen Stadt sowie 400 gefälschte Rezensionen der gleichen 20 Hotels, die von über Amazon rekrutierten Crowd-Workern erstellt wurden.

Die insgesamt 800 Rezensionen sind zu gering, um heutige Klassifikationsverfahren effizient zu trainieren. Zudem wurden ausschließlich positive Rezensionen mit einer Länge von über 150 Zeichen berücksichtigt, was keiner realen Verteilung entspricht [44].

Sandulescu und Ester [55] erstellen in ihrer Arbeit neben dem Datensatz von Ott et al. [44] zusätzlich einen Yelp-Datensatz, bei dem alle Rezensionen vom Unternehmen selbst vorab mittels eines internen Filters in „recommended“ und „not recommended“ unterteilt wurden. Allerdings wird nicht direkt offengelegt, auf welcher Grundlage diese Klassifizierung erfolgt, beispielsweise ob sie auf dem Verhalten der Rezensenten oder auf der Analyse der verfassten Texte basiert [56]. Zu dem wurde ein Datensatz von Trustpilot verwendet, bei dem die Rezensionen ebenso vom Unternehmen bereits gefiltert wurden.

Salminen et al. [53] veröffentlichten einen in Satzlänge, Bewertung und Produktkategorie balancierten Datensatz, bestehend aus 20.000 von GPT-2 generierten sowie 20.000 echten Amazon-Produktrezensionen.

Siegel [56] untersuchte mit Studierenden der Hochschule Darmstadt Opinion Spam im deutschsprachigen Amazon-Portal und erstellte einen Korpus mit mehreren Hundert Beispielen. Dieser bietet zwar eine solide Grundlage für weitergehende Untersuchungen, ist für den Einsatz von automatischen Lernverfahren jedoch zu klein.

Detektion von Opinion Spam

Laut des Systematic Literature Review zu Machine-based Opinion Spam [47] sind die am häufigsten eingesetzten Klassifikatoren zur Erkennung von Review Spam in Datensätzen ohne maschinell generierte Rezensionen statisti-

sche Verfahren wie Support Vector Machines [13, 50] oder Naive Bayes [13, 38]. An dritter Stelle folgt der K-Nearest-Neighbor-Algorithmus [13].

Salminen et al. [53] nutzten für die Klassifizierung ihrer von GPT-2 generierten Rezensionen ein feinjustiertes RoBERTa-Modell.

Liyanage et al. [31] verglichen verschiedene State-of-the-Art-Klassifikatoren, die auf der Transformer-Architektur basieren. Unter den untersuchten Modellen erzielte BERT die besten Ergebnisse bei der Identifikation von Rezensionen, die mithilfe verschiedener LLMs auf Basis des Opinion-Spam-Korpus von Ott et al. [44] als Seed generiert wurden.

Bisher existieren nur wenige Ansätze, bei denen Transformermodelle wie BERT oder GPT zur Erkennung von Opinion Spam eingesetzt wurden. Gerade in diesem Anwendungsbereich ist der Einsatz solcher Modelle jedoch besonders relevant, da sich häufig ausschließlich der Text ohne zusätzliche Metadaten analysieren lässt, was die Erkennung durch weniger komplexe Verfahren erschwert.

DATENSATZERSTELLUNG

Die Erstellung eines qualitativ hochwertigen Datensatzes stellt den zentralen methodischen Bestandteil dieser Arbeit dar. Ziel ist es, einen ausgewogenen Korpus aus echten und generierten Rezensionen zu entwickeln, der als Grundlage für die Analyse und Klassifikation von Opinion Spam dient. Dieses Kapitel beschreibt die einzelnen Schritte des Erstellungsprozesses.

4.1 EXPERIMENTELLE UMGEBUNG

Die praktische Umsetzung der Datensatzerstellung erfolgte in einer cloudbasierten Entwicklungsumgebung unter Verwendung von Google Colab und einer NVIDIA A100 GPU. Diese wurde insbesondere für das Training des [RoBERTa](#)-Modells benötigt. Die Implementierung wurde vollständig in Python realisiert, da sich die Programmiersprache im Bereich des Machine Learning und [NLP](#) etabliert hat und ein umfangreiches Ökosystem spezialisierter Bibliotheken für Datenanalyse, Textverarbeitung und maschinelles Lernen bietet. Für die Interaktion mit den Modellen kamen unter anderem die Bibliotheken [OpenAI](#), [sklearn](#), [transformers](#), [pandas](#) und [torch](#) zum Einsatz.¹

4.2 PIPELINE

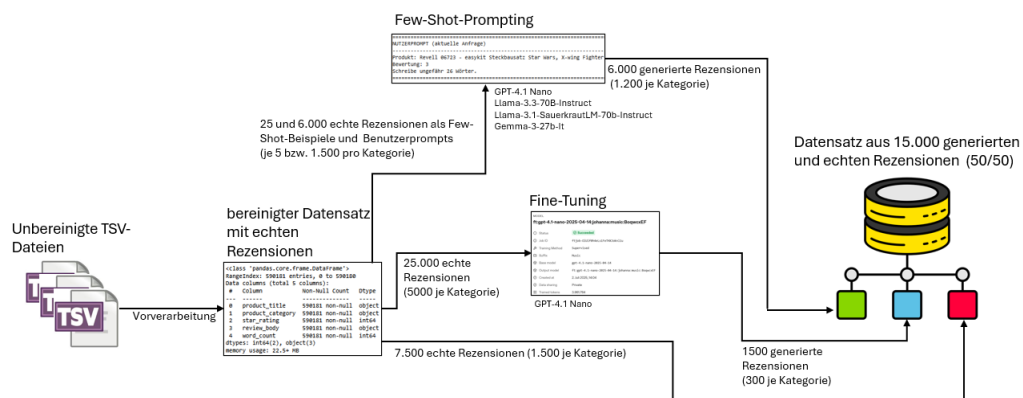


Abbildung 4.1: Prozess der Datensatzerstellung.

Abbildung 4.1 zeigt den Ablauf des Prozesses zur Erstellung des Datensatzes. Im ersten Schritt erfolgte die Überführung der ursprünglich in TSV-Dateien vorliegenden Rezensionen in ein geeignetes Format sowie deren Be-

¹ Der vollständige Code zur Datensatzerstellung sowie der Datensatz sind unter <https://code.fbi.h-da.de/stbojohan/masterarbeit> verfügbar.

reinigung (vgl. Kapitel 4.3). Auf Basis der bereinigten Daten wurde anschließend unter Anwendung verschiedener Techniken ein qualitativ hochwertiger Datensatz aus echten und generierten Rezensionen erstellt. Für die Generierung mittels Few-Shot-Prompting kamen 25 Rezensionen aus dem vorverarbeiteten Datensatz als Few-Shot-Beispiele zum Einsatz, wobei jeweils fünf Rezensionen pro Kategorie herangezogen wurden. 6.000 echte Rezensionen wurden für die Benutzer-Prompts verwendet (vgl. Kapitel 4.4.1). Für jede Kategorie wurden mit vier unterschiedlichen LLMs insgesamt 1.200 Rezensionen generiert. Darüber hinaus dienten 5.000 Rezensionen pro Kategorie zum Fine-Tuning von fünf verschiedenen GPT-Modellen, die jeweils 300 Rezensionen erzeugten (vgl. Kapitel 4.4.2). Ergänzend dazu wurden 7.500 echte Rezensionen aus dem bereinigten Datensatz entnommen, sodass der finale Datensatz aus insgesamt 15.000 Rezensionen besteht. Dieses ausgeglichene Klassenverhältnis wurde gewählt, da es ein effektiveres Training der Klassifikationsmodelle ermöglicht. Insbesondere können die Modelle unter diesen Bedingungen gleichwertig Merkmale beider Klassen erlernen, ohne durch eine dominante Klasse verzerrt zu werden.

4.3 DATENVORVERARBEITUNG

Sowohl für die echten als auch für die generierten Rezensionen wurde der Datensatz *The Multilingual Amazon Reviews Corpus* von Keung et al. [26] verwendet. Die Daten wurden am 13.12.2017 abgerufen. Der Datensatz besteht aus Amazon-Produktrezensionen zwischen 1997 und 2015. Die Rezensionen stammen aus einer Zeit, in der LLMs noch nicht existierten. Daher kann ausgeschlossen werden, dass maschinell generierte Rezensionen im Datensatz vorhanden sind. Allerdings ist nicht auszuschließen, dass von Menschen verfasster Opinion Spam enthalten ist. Der Rohdatensatz umfasst etwa 679.000 Rezensionen.

Um die Qualität der Daten sicherzustellen, wurden zunächst unvollständige Einträge entfernt. Diese umfassten alle Rezensionen, bei denen das Feature `review_date` einen Nullwert aufwies. Aufgrund der großen Datenmenge war es nicht erforderlich, auf spezielle Techniken zur Datenimputation zurückzugreifen. Stattdessen wurden ausschließlich Rezensionen berücksichtigt, deren Metadaten vollständig vorlagen. Erste Analysen zeigten zudem, dass der Datensatz auch englischsprachige Rezensionen enthielt. Zur Identifikation der jeweiligen Sprache wurde die Python-Bibliothek *language-detect* [8] eingesetzt, sodass ausschließlich deutschsprachige Rezensionen (92,6 %) im Datensatz verblieben.

Ein weiterer Bereinigungsprozess bestand im Entfernen mehrfach vorkommender Rezensionen. Identische Rezensionstexte für verschiedene Produkte einzusetzen, ist ein typisches Vorgehen von Spammern. Jindal und Liu [23], die als erste das Problem des Opinion Spam bei Produktrezensionen beschrieben, legten ihren Schwerpunkt auf die Identifikation von duplizierten Rezensionen. Das Eliminieren von Duplikaten betraf etwa 20.000 Rezensionen. Zur Bereinigung der Rezensionen wurden zudem HTML-

und Steuerzeichen entfernt, darunter Zeilenumbrüche, Anführungszeichen-Codierungen, doppelte Schrägstriche sowie Backslashes.

Abbildung 4.2 zeigt die Verteilung der Wortlängen der Rezensionen. Nur etwa 3 % der Rezensionen enthalten mehr als 500 Wörter. Da sowohl beim Fine-Tuning als auch beim Few-Shot-Prompting echte Rezensionen als Trainings- bzw. Prompt-Daten verwendet werden und das Generieren vieler Tokens sehr ressourcenintensiv ist [25], wurden nur Rezensionen mit bis zu 500 Wörtern beibehalten. Zudem wurden nur Rezensionen mit mindestens fünf Wörtern behalten, da kürzere Texte zu wenig Informationen beinhalten. Lediglich 0,01 % der Rezensionen hatten weniger als fünf Wörter.



Abbildung 4.2: Verteilung der Rezensionenlängen.

Aus den insgesamt 32 Kategorien des Datensatzes wurden die Kategorien *Books*, *Mobile Apps*, *Music*, *PC* und *Toys* ausgewählt. Die ursprünglich größte Kategorie, *Video*, wurde bewusst ausgeschlossen, da sie überwiegend Rezensionen zu Filmen und Serien enthält und somit thematisch nicht zur Produktfokussierung des Datensatzes passt. Für jede Kategorie wurden jeweils 5.000 Rezensionen mittels stratifizierter Stichprobenziehung extrahiert, um die ursprüngliche Verteilung von Wortanzahl und Sternebewertung zu erhalten. Bei dieser Methode wird die Grundgesamtheit in Schichten aufgeteilt, die gemeinsame Merkmale aufweisen, und anschließend werden aus diesen Schichten Stichproben entnommen [46]. Von diesen 5.000 extrahierten Rezensionen dienen 30 % als echte Rezensionen im finalen Datensatz, während aus den restlichen 70 % zufällig pro Kategorie fünf Rezensionen als Few-Shot-Prompts und 1.200 als Benutzerprompts für die Generierung der synthetischen Rezensionen verwendet wurden. Für das Fine-Tuning wurden pro Kategorie die gesamten 5.000 Rezensionen verwendet (vgl. Abb. 4.1). Da in dieser Arbeit ausschließlich die inhaltliche Erkennung von *Spam Reviews* im Vordergrund steht, wurden sämtliche in den Rohdaten enthaltene Metadaten entfernt. Nach der Eliminierung irrelevanter Features ergibt sich der in Tabelle 4.1 dargestellte Aufbau des bereinigten Datensatzes.

Feature	Beschreibung
product_title	Name des Produkts
product_category	Kategorie der Produktrezension
review_body	Inhalt der Produktrezension
star_rating	Sternebewertung der Produktrezension
word_count	Rezensionslänge in Wörtern

Tabelle 4.1: Übersicht der verwendeten Features.

4.4 REZENSIONSGENERIERUNG

Die Generierung von Rezensionen stellt den zentralen Bestandteil der Datensatzerstellung dar und verfolgt das Ziel, künstliche Texte zu erzeugen, die in Stil und Inhalt realen Bewertungen möglichst nahekommen. Im Folgenden wird das methodische Vorgehen zur Erstellung der generierten Rezensionen beschrieben. Grundlage bildet der Einsatz vier verschiedener **LLMs**. Die Verwendung unterschiedlicher **LLMs** ermöglicht es, die Detektion von Opinion Spam nicht auf die Charakteristika eines einzelnen Modells zu beschränken, sondern ein breiteres Spektrum generierter Texte abzudecken. Zur Umsetzung wurden Few-Shot-Prompting und Fine-Tuning verwendet. Diese Entscheidung wurde getroffen, um unterschiedliche Ansätze abzudecken und deren Einfluss auf die Erkennbarkeit künstlicher Texte zu untersuchen. Das folgende Kapitel beschreibt die Implementierung dieser Verfahren.

4.4.1 Few-Shot-Prompting

Die Umsetzung der Rezensionsgenerierung mittels Few-Shot Prompting erfolgte unter Verwendung der Python-Bibliothek OpenAI [39]. Mit Ausnahme des **GPT**-Modells wurden alle verwendeten Modelle durch die *Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG)* bereitgestellt und über eine API zugänglich gemacht. Jedes der Modelle (*GPT-4.1 Nano*, *Llama-3.1-SauerkrautLM-70b-Instruct*, *Llama-3.3-70B-Instruct* und *Gemma-3-27b-It*) generierte pro Kategorie jeweils 300 Rezensionen.

Die Prompts für die Modelle besaßen die folgende Struktur und den folgenden Inhalt:

1. **Systemprompt** – allgemeine Instruktionen zum Schreibstil:
 - alltagssprachlich und natürlich
 - keine unnötigen Wiederholungen
 - keine formelle Ausdrucksweise
 - Beachtung eines vorgegebenen Wortumfangs
2. **Few-Shot-Beispiele** aus den echten Daten:

- Input: Produktname und Sternebewertung
- Output: Originalrezension aus dem Datensatz

3. Benutzerprompt – neue Anfrage an das Modell mit:

- Produktname
- Sternebewertung
- gewünschte Wortanzahl

Für die Few-Shot-Beispiele wurden mit einem festgelegten Seed pro Kategorie jeweils fünf echte Rezensionen als Beispiele aus dem vorverarbeiteten Datensatz ausgewählt. In verschiedenen Arbeiten wurde die Anzahl der Beispiele im Bereich von eins bis acht gewählt (vgl. [7, 66]). Da in dieser Arbeit fünf Rezensionen mit bis zu fünf Sternen generiert werden, wird $k = 5$ gewählt, sodass für jedes Rating eine Beispielrezension verwendet wird. Als Input dienten Produktname und Sternebewertung, während der Inhalt der Rezension als Output angegeben wurde.

Der Benutzerprompt, der die eigentliche Instruktion an das Modell darstellt, enthielt analog zu den Few-Shot-Beispielen Produktname, Sternebewertung sowie eine geforderte ungefähre Wortanzahl. Hierfür wurden ebenfalls zufällig die entsprechenden Attributwerte der echten Rezensionen aus dem vorverarbeiteten Datensatz übernommen. Um eine zu starke Musterbildung zu vermeiden, wurde für jedes Modell ein eigener Seed verwendet, sodass die Modelle nicht zu denselben Produkten Rezensionen generierten. Auf diese Weise erhielten alle vier Modelle zwar identische Few-Shot-Beispiele, jedoch für die jeweils 1.500 zu generierenden Rezensionen unterschiedliche Benutzerprompts. Der verwendete API-Aufruf `client.chat.completions.create(...)` von OpenAI [39] erwartet einen Prompt mit dem Aufbau wie in Listing 4.1.

```
[
  {
    "role": "system",
    "content": "Du bist ein Käufer oder eine Käuferin, der/die ehrliche,
      natürlich klingende Amazon-Produktrezensionen schreibt."
  },
  {
    "role": "user",
    "content": "Produkt: Kuschelteddy
      Bewertung: 5"
  },
  {
    "role": "assistant",
    "content": "Der Teddy ist unglaublich weich und süß. Meine Tochter
      schläft jede Nacht mit ihm ein und will ihn gar nicht mehr
      hergeben."
  },
  {
    "role": "user",
```

```

    "content": "Produkt: Spielzeugauto
    Bewertung: 2"
  },
  {
    "role": "assistant",
    "content": "Das Auto sieht zwar schick aus, aber nach kurzer Zeit
    gingen die Räder kaputt."
  },
  {
    "role": "user",
    "content": "Produkt: Bauklötze
    Bewertung: 4
    Schreibe ungefähr 30 Wörter."
  }
]

```

Listing 4.1: Beispiel für Prompt mit zwei Few-Shot-Beispielen.

Dieser besitzt jeweils die zwei Felder Rolle (*role*) und Inhalt (*content*). Die Rolle definiert die Perspektive der Nachricht und wie das Modell sie interpretieren soll. Es gibt drei unterschiedliche Rollen:

system: Definiert allgemeine Instruktionen zum Schreibstil und gewünschtem Verhalten des Modells.

user: Enthält die Eingabeaufforderung des Nutzens.

assistant: Enthält Beispielantworten, die das Modell als Referenz nutzt.

Durch die Abfolge von Systemprompt, den User- und Assistant-Paaren (Few-Shot-Beispiele) sowie der abschließenden User-Nachricht (Benutzerprompt) entsteht der vollständige Prompt. Dieser vermittelt dem Modell sowohl die stilistischen Rahmenbedingungen als auch konkrete Beispiele für Eingabe-Ausgabe-Paare und die eigentliche Arbeitsanweisung, auf die das Modell mit einer neuen Assistant-Nachricht in Form einer generierten Rezension reagiert. Zur Erhöhung der Kreativität der generierten Rezensionen wurde die Decoding-Strategie Top-p-Sampling mit einem p-Wert von 0,9 eingesetzt. Sie bietet eine dynamische Alternative zu deterministischen Verfahren wie Beam Search und dem starren Top-k-Sampling. Im Anhang [A.1](#) ist ein vollständiger Beispiel-Prompt für die Generierung einer Rezension der Kategorie *Toys* zu finden.

4.4.2 *Fine-Tuning-basierte Generierung*

Das GPT-Modell wurde neben dem Few-Shot-Prompting auch für das Fine-Tuning verwendet. Dafür wurde die von OpenAI bereitgestellte Funktionalität benutzt. Für jede der fünf ausgewählten Kategorien *Books*, *Mobile Apps*, *Music*, *PC* und *Toys* wurde ein separates Modell finegetuned. Grundlage hierfür war eine für jede Kategorie vorbereitete JSON-Datei, die die entsprechenden Trainingsdaten enthielt. Jeder Eintrag in der Datei bestand aus einer standardisierten Nachrichtenstruktur mit den drei Rollen analog zu der

Struktur beim [Few-Shot-Prompting](#). Listing 4.2 zeigt beispielhaft, wie ein Eintrag aus dem Trainingsdatensatz im JSON-Format aussah. Ein Beispiel aus den realen Daten ist im Anhang [A.2](#) dargestellt.

```
[
  {
    "role": "system",
    "content": "Du bist ein Käufer oder eine Käuferin, der/die ehrliche,
      natürlich klingende Amazon-Produktrezensionen schreibt."
  },
  {
    "role": "user",
    "content": "Produkt: Handyhülle
      Bewertung: 1"
  },
  {
    "role": "assistant",
    "content": "War nach einer Woche vergilbt und musste weggeworfen
      werden."
  }
]
```

Listing 4.2: Beispiel Rezension im JSON-Format für Fine-Tuning.

Zu der von OpenAI bereitgestellten Funktionalität gehört eine Fine-Tuning-API sowie ein Webinterface. Dieses Interface bietet eine übersichtliche Darstellung des Trainingsfortschritts und erlaubt die Überwachung zentraler Metriken wie dem Trainingsverlust und der *Train Mean Token Accuracy*. Die Visualisierung erfolgt dabei in Form interaktiver Kurven, die automatisch geglättet dargestellt werden, um die Lernrends klarer zu erkennen.

Die Kosten für das Fine-Tuning von *GPT-4.1 Nano* betragen 1,5 US-Dollar pro eine Millionen Token. Pro Kategorie wurde das Modell jeweils mit etwa vier Millionen Token finegetuned, was Gesamtkosten von rund 30 US-Dollar ergab. Die Kosten für die Generierung der 1.500 Rezensionen durch die fünf feinabgestimmten Modelle sowie der weiteren 1.500 Rezensionen des *GPT*-Modells mittels Few-Shot-Prompting beliefen sich insgesamt auf zusätzlich einen US-Dollar.

Das Fine-Tuning der Modelle erfolgte unter Verwendung der von OpenAI empfohlenen Standard-Hyperparameter. Die Anzahl der Epochen wurde auf drei festgelegt. Eine Epoche bezeichnet einen vollständigen Durchlauf des Modells über den gesamten Trainingsdatensatz. Ergänzend dazu wurde eine Batchgröße von zehn verwendet. Diese gibt an, wie viele Trainingsbeispiele gleichzeitig in einem Optimierungsschritt verarbeitet werden. Bei 5000 Trainingsdaten heißt das, dass 500 Trainingsschritte einer Epoche entsprechen, sodass bei drei Epochen insgesamt 1500 Gewichtsadjustierungen vorgenommen wurden. Nach jeder Anpassung wurden der Trainingsverlust und die *Train Mean Token Accuracy* berechnet.

Die Wahl einer kleinen Batchgröße führt dazu, dass die Modellparameter häufiger aktualisiert werden. Dies fördert eine stabilere Konvergenz, verringert das Risiko einer lokalen Überanpassung und führt zugleich zu dichter

teren und stärker verrauschten Kurvenverläufen. Ein weiterer zentraler Hyperparameter war der Learning-Rate-Multiplikator, der auf 0,1 gesetzt wurde. Die Reduzierung der ursprünglichen Lernrate führt zu kleineren Anpassungsschritten. Dies hat eine langsamere, aber stabilere Annäherung an die Zielverteilungen der Trainingsdaten zur Folge.

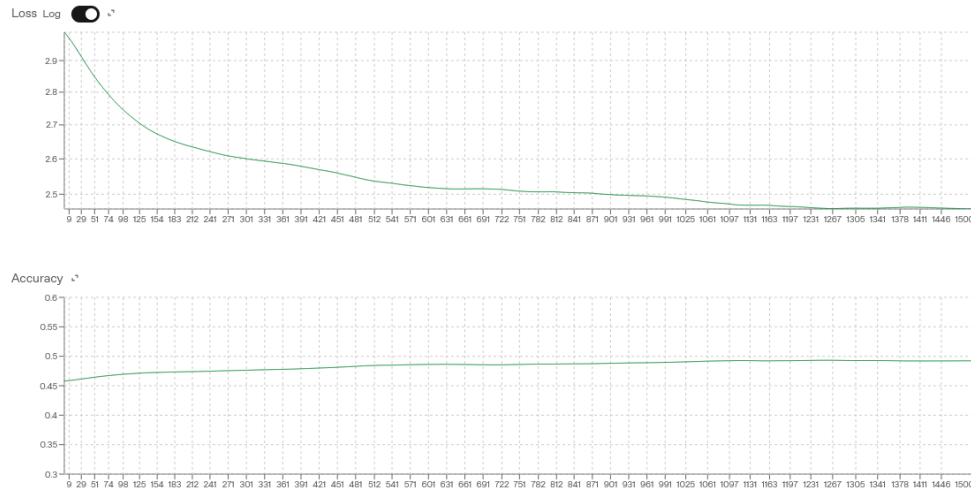


Abbildung 4.3: Loss und Accuracy Fine-Tuning *Music*.

Abbildung 4.3 zeigt den Verlauf des Trainingsverlustes und die *Train Mean Token Accuracy* für das Fine-Tuning des GPT-Modells für die Kategorie *Music*. Der Trainingsverlust im Kontext des Fine-Tunings ist der, wie bereits im Kapitel 2.3.1 zur logistischen Regression eingeführte, Kreuzentropieverlust (auch *Log Loss*) (vgl. Formel 2.10). Er misst, wie gut das Modell Wahrscheinlichkeiten für die richtigen Tokens vorhersagt. Die *Train Mean Token Accuracy* beschreibt den Anteil der Tokens, bei denen das Modell das richtige Token mit der höchsten Wahrscheinlichkeit gewählt hat [36]. Um die Lesbarkeit und Interpretierbarkeit zu verbessern, wurden die Kurven mittels der Funktionalität von OpenAI geglättet. Die Originalverläufe der Fine-Tunings aller Kategorien ohne Glättung befinden sich der Vollständigkeit halber im Anhang A.3.

Der Trainingsverlust zeigt einen kontinuierlichen und gleichmäßigen Abfall von etwa 3,0 auf 2,45, was auf einen stabilen Lernfortschritt des Modells hinweist. Nach ungefähr 1100 Trainingsschritten nimmt das Sinken des Verlusts stark ab, was auf eine weitgehende Konvergenz des Modells schließen lässt. Ein erneuter Anstieg, der auf Overfitting hindeuten würde, ist nicht zu beobachten. Die Verläufe der anderen Kategorien sind analog (vgl. Anhang A.3) und weisen nur minimale Unterschiede auf.

Eine häufig verwendete Metrik, um die Fähigkeit eines LLMs, das nächste Wort in einer Sequenz vorherzusagen, ist die Perplexität. Sie definiert sich Exponent des durchschnittlichen negativen Log-Likelihoods [2].

Für den finalen Trainingsverlust $\text{LogLoss} = 2,418$ ergibt sich beim Modell in der Kategorie *Music* eine Perplexity von

$$\text{PPL} = e^{2.418} \approx 11,2 \quad (4.1)$$

was bedeutet, dass das Modell im Schnitt etwa 11 Tokens als plausible Fortsetzung in Betracht zieht.

Der Verlauf der *Train Mean Token Accuracy* [40] zeigt einen leichten, aber stetigen Anstieg von etwa 0,45 auf 0,50, bevor sie konvergiert. Dies deutet darauf hin, dass das Modell im Laufe des Fine-Tunings moderate, aber stabile Leistungsverbesserungen erzielt hat.

Tabelle 4.2 zeigt die finalen Trainingsverluste sämtlicher untersuchter Kategorien. Die Ergebnisse der Kategorien *Music*, *PC* und *Toys* sind sehr ähnlich. *Mobile Apps* hat mit 14,15 die schlechteste und *Books* mit 8,84 die beste Perplexity. Im Gegensatz dazu hat das Modell der Kategorie *Mobile Apps* die höchste *Train Mean Token Accuracy*.

Zusätzlich wurde initial versucht, das Modell *Mistral-7B* mittels *LoRA* selbst finezutunen. In mehreren Experimenten zeigte sich jedoch, dass die resultierende Textqualität deutlich unter den Ergebnissen des verwendeten *GPT-4.1 Nano*-Modells lag. Die generierten Rezensionen waren semantisch weniger kohärent, enthielten häufiger repetitive Sequenzen und wirkten insgesamt weniger realistisch. Aufgrund dieser unzureichenden Ergebnisse wurde der Ansatz verworfen und nicht weiter verfolgt.

Kategorie	finaler Trainingsverlust	finale Accuracy	Perplexity
Books	2,18	0,492	8,84
Mobile Apps	2,65	0,467	14,15
Music	2,42	0,529	11,24
PC	2,41	0,467	11,13
Toys	2,51	0,477	12,30

Tabelle 4.2: Fine-Tuning-Evaluation.

5.1 QUALITÄTSANALYSE DES DATENSATZES

Um die Aussagekraft des erstellten Datensatzes sicherzustellen, ist zunächst eine Analyse seiner Qualität erforderlich. Dabei soll untersucht werden, inwiefern sich die generierten Rezensionen von den echten unterscheiden. Neben einer deskriptiven statistischen Betrachtung, beispielsweise der Verteilung von Wortanzahl, Sternebewertungen und der Anteile der zugrundeliegenden Modelle, wird zudem eine Metrik zur Textdiversität herangezogen. Auf diese Weise lässt sich einschätzen, ob generierte Texte auffällige Muster aufweisen oder in ihrer Struktur und Ausdrucksweise mit echten Rezensionen vergleichbar sind.

5.1.1 Statistische Analyse

Der erstellte Datensatz umfasst insgesamt 15.000 Rezensionen, von denen jeweils 50 % echte und 50 % generierte Texte sind. Für die Generierung wurden die vier verschiedenen LLMs *GPT-4.1 Nano*, *Llama-3.1-SauerkrautLM-70b-Instruct*, *Llama-3.3-70B-Instruct* und *Gemma-3-27b-It* eingesetzt. Während alle Modelle mittels Few-Shot-Prompting genutzt wurden, wurde das *GPT*-Modell zusätzlich finegetuned (vgl. Abb. 5.1). Jedes Modell trug dabei mit 1.500 Rezensionen zum Datensatz bei, sodass eine ausgewogene Verteilung der Quellen gewährleistet ist. Zudem ist die Anzahl der Kategorien gleichverteilt. Jede der fünf Kategorien enthält 3.000 Rezensionen. Dies ist relevant, um bei der Klassifikation der generierten Rezensionen eine Vergleichbarkeit sicherzustellen.

Ein zentrales Ziel war es, sicherzustellen, dass sich die echten und generierten Rezensionen in ihrer Grundverteilung möglichst wenig unterscheiden. Dies betrifft sowohl die Auswahl der echten Rezensionen aus dem ursprünglichen Datensatz mit echten Amazon-Rezensionen, als auch die Rezensionen, die als Prompts für die Modelle genutzt wurden. In den Prompts wurden jeweils die Wortanzahl, Sternebewertung und der Titel mitgegeben, sodass die generierten Rezensionen dieselben strukturellen Eigenschaften aufweisen wie die echten Rezensionen (vgl. Kapitel 4.4).

Abbildung 5.2 zeigt, dass die Verteilungen der Sternebewertungen und Textlängen zwischen den echten und den generierten Rezensionen insgesamt ähnlich ausfallen. Da die für die Benutzerprompts verwendeten Rezensionen zufällig aus den stratifizierten Stichproben gezogen wurden, ergeben sich kleinere Abweichungen in den Verteilungen. Zudem fordert der Prompt die Einhaltung der Wortanzahl nur näherungsweise, wodurch Abweichungen bei den Längen der Rezensionen erklärbar sind. Durch die weitgehend

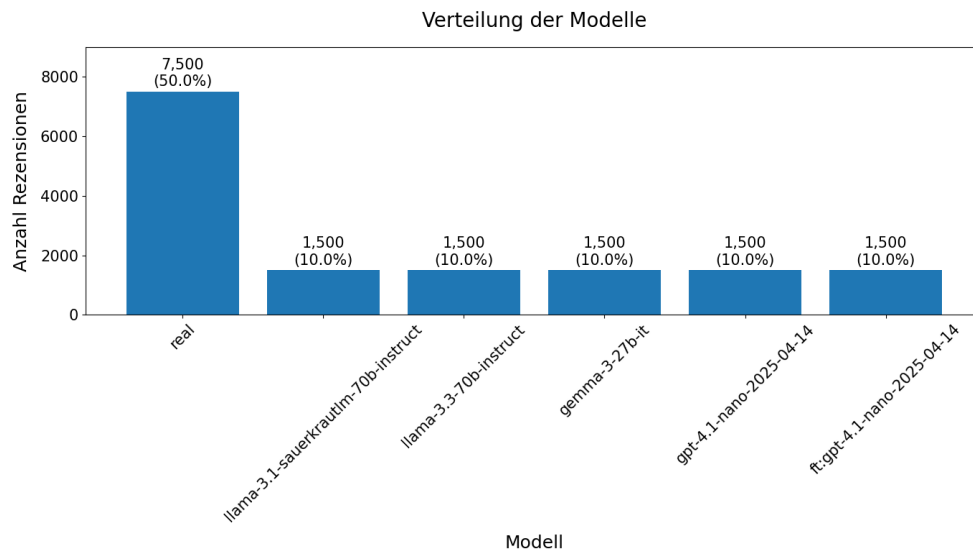


Abbildung 5.1: Verteilung der Modelle im Datensatz.

übereinstimmenden Verteilungen von Textlänge, Sternebewertung und Kategorie zwischen echten und generierten Rezensionen lassen sich Unterschiede in den späteren Klassifikationsergebnissen nicht auf triviale oberflächliche Merkmale zurückführen. Die Beobachtungen basieren somit tatsächlich auf sprachlichen und inhaltlichen Charakteristika der Texte.

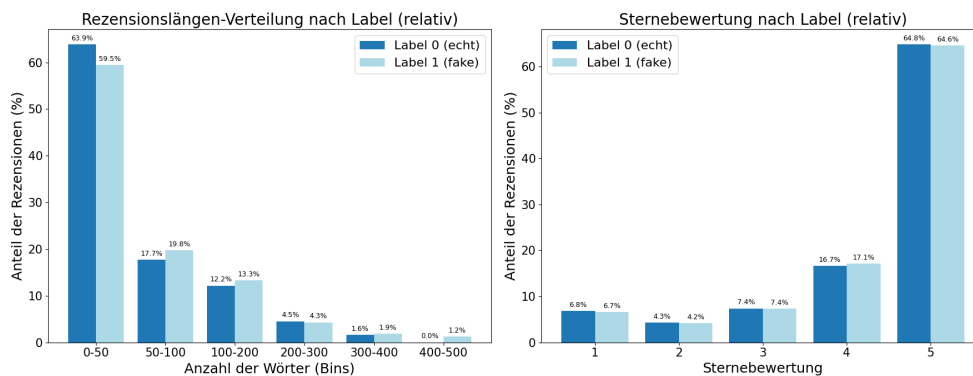


Abbildung 5.2: Verteilungen der echten und generierten Rezensionen im Bezug auf Rezensionslänge und Sternebewertung.

5.1.2 Sprachliche Vielfalt

Neben der statistischen Betrachtung ist für die Bewertung des Datensatzes entscheidend, die sprachliche Qualität der Rezensionen zu untersuchen. Dabei wird analysiert, inwiefern sich echte und generierte Texte hinsichtlich ihrer Diversität unterscheiden.

Zur Bewertung der sprachlichen Vielfalt der Rezensionen wurde Self-BLEU [71] eingesetzt. Die Metrik berechnet die Ähnlichkeit einer Rezension mit Rezensionen einer zufällig ausgewählten Teilmenge. Niedrigere Werte deu-

Modell	BLEU	BLEU-1	BLEU-2	BLEU-3	BLEU-4
GPT-4.1 Nano	1,83	26,91	4,00	1,59	0,81
GPT-4.1 Nano (finegetuned)	1,26	24,22	2,38	0,95	0,49
Lama-3.1-SauerkrautLM-70b-Instruct	1,47	24,09	3,27	1,20	0,63
Llama-3.3-70B-Instruct	1,76	24,12	3,81	1,63	0,86
Gemma-3-27b-It	1,52	23,86	3,20	1,25	0,66
alle Modelle	1,35	22,96	2,86	1,12	0,58
echte Rezensionen	0,91	17,65	1,65	0,75	0,38

Tabelle 5.1: BLEU- und n-Gramm-Präzisionen der generierten und echten Rezensionen.

ten auf eine größere sprachliche Diversität hin, während höhere Werte homogenere und stärker wiederholende Strukturen bedeuten. Für die Berechnung wurden insgesamt 1.000 Rezensionen zufällig ausgewählt und mit jeweils 200 Referenz-Rezensionen verglichen. Anschließend wurde der Durchschnittswert berechnet.

Die Ergebnisse in Tabelle 5.1 zeigen, dass die echten Rezensionen die größte Vielfalt aufweisen. Das feinabgestimmte GPT-Modell (*GPT-4.1 Nano*) erreicht mit 1,26 den niedrigsten Wert unter den Modellen und generiert somit die diversesten Texte. Dagegen besitzen die Rezensionen der Modelle *GPT-4.1 Nano* (Few-Shot-Prompting) mit 1,83 und *Llama-3.3-70B-Instruct* mit 1,76 die höchsten Werte, was auf eine geringere Varianz in den Formulierungen hinweist.

Am zweitbesten schneidet die Gruppe der kombinierten Modelle ab, was darauf hindeutet, dass die unterschiedlichen Sprachmodelle bei der Generierung jeweils abweichende Wörter und Phrasen bevorzugen und sich ihre sprachlichen Muster ergänzen. Auffällig ist zudem, dass die beiden GPT-Modelle am häufigsten dieselben Wörter verwenden (BLEU-1). Das nicht feinabgestimmte Modell *GPT-4.1 Nano* sowie *Llama-3.3-70B-Instruct* verwenden am häufigsten gleiche Phrasen (BLEU-2 - BLEU-4). Im Anhang A.4 befindet sich zur Veranschaulichung jeweils eine generierte Rezension von jedem Modell der Kategorie *Music*.

5.2 KLASSIFIKATION DER REZENSIONEN

Nachdem im vorangegangenen Abschnitt die Qualität des Datensatzes untersucht wurde, soll nun überprüft werden, inwiefern sich die generierten Rezensionen automatisiert von den realen unterscheiden lassen. Ziel ist es, zu analysieren, wie einfach die generierten Rezensionen von zwei verschiedenen Modellen als solche klassifiziert werden können. Als Klassifikatoren wurden eine logistische Regression und RoBERTa verwendet. Der Vergleich zwischen einem linearen Modell und einem Transformer ermöglicht eine

fundierte Bewertung, inwiefern komplexere sprachliche Muster zur Detektion generierter Rezensionen beitragen.

5.2.1 Modelltraining

Logistische Regression

Für die Klassifikation mit der logistischen Regression wurden die Rezensionstexte zunächst in eine numerische Repräsentation überführt. Hierzu kam eine **TF-IDF**-Vektorisierung zum Einsatz, bei der jedem Wort im Korpus ein Gewicht zugeordnet wird. Im Anschluss wurde die logistische Regression auf diesen Vektoren trainiert. Dabei wird die Kreuzentropie-Verlustfunktion mittels des Verfahrens des **SGD** minimiert. Um eine Überanpassung zu vermeiden, wird die L2-Regularisierung eingebaut (Kapitel vgl. 2.3.1).

Die Auswahl geeigneter Hyperparameter (Anzahl zu berücksichtigender Wörter bei **TF-IDF** und Anzahl der Iterationen bei der logistischen Regression) erfolgte mithilfe einer Grid Search in Kombination mit 5-facher Kreuzvalidierung. Dabei wird das Trainingsset in mehrere Teilmengen aufgeteilt, sodass das Modell wiederholt auf unterschiedlichen Kombinationen von Trainings- und Validierungsdaten trainiert und evaluiert wird. Dadurch lässt sich die Generalisierbarkeit des Modells besser beurteilen und das Risiko einer Überanpassung reduzieren. Der Datensatz wurde im Verhältnis 80% Training zu 20% Test aufgeteilt.

Bei der logistischen Regression erhält jedes Wort im Vokabular einen Koeffizienten, der seine Bedeutung für die Klassifikation ausdrückt. Positive Koeffizienten weisen darauf hin, dass das Auftreten eines Wortes häufiger mit authentischen Rezensionen verknüpft ist, während negative Koeffizienten darauf hindeuten, dass ein Wort eher in generierten Rezensionen vorkommt. Tabelle 5.2 zeigt die relevantesten Wörter für eine Klassifikation.

Klasse 1 (Echt)		Klasse 0 (Fake)	
Wort	Koeffizient	Wort	Koeffizient
der	2,87	Die	-8,42
doch	2,13	Ein	-5,60
als	2,12	Sehr	-4,97
nun	2,08	Klare	-4,62
diesen	2,08	okay	-3,98
am	2,01	Ich	-3,78
bzw	2,00	perfekt	-3,75
nicht	1,99	empfehlenswert	-3,74
schon	1,86	aber	-3,68
im	1,85	Perfekt	-3,62

Tabelle 5.2: Top 10 Wörter nach Koeffizienten für logistische Regression.

Die Analyse der Koeffizienten der logistischen Regression offenbart klare sprachliche Unterschiede, die für die Zuordnung zu den jeweiligen Klassen entscheidend sind. Wörter mit positiven Koeffizienten sind überwiegend Funktionswörter wie *der*, *doch*, *als* oder *im*. Diese deuten darauf hin, dass authentische Rezensionen in einer eher alltäglichen, unauffälligen Sprache verfasst sind, die typische grammatische Strukturen und Nuancen enthält. Auch Begriffe wie *nicht* oder *schon* weisen auf eine differenzierte Ausdrucksweise hin, wie sie für natürlich verfasste Texte charakteristisch ist.

Im Gegensatz dazu weisen gefälschte Rezensionen stark negative Koeffizienten für meist positiv evaluative Wörter wie *Perfekt*, *Sehr* oder *empfehlenswert* auf. Diese deuten auf eine übersteigerte Sprache hin, die in den echten Rezensionen seltener vorkommt. Zudem tritt das Pronomen *Ich* häufiger in gefälschten Rezensionen auf, was darauf hindeutet, dass die LLMs versuchen, durch eine betonte Ich-Perspektive Authentizität zu simulieren. Dass die Wörter mit hohen negativen Koeffizienten überwiegend großgeschrieben sind, deutet zudem darauf hin, dass die generierten Rezensionen häufig mit den gleichen Wörtern beginnen.

RoBERTa

Als zweites wurde RoBERTa für die Klassifikation verwendet. Für den vorliegenden Anwendungsfall wurde das Modell in einem Fine-Tuning-Prozess an die binäre Klassifikationsaufgabe angepasst. Dabei wurden die trainierbaren Parameter auf Grundlage der Trainingsdaten aktualisiert, um die Fähigkeit des Modells zur Unterscheidung zwischen echten und generierten Rezensionen zu optimieren.

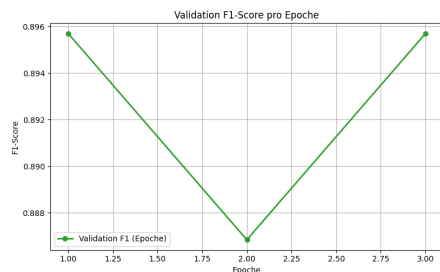


Abbildung 5.3: F1-Validation pro Epoche.

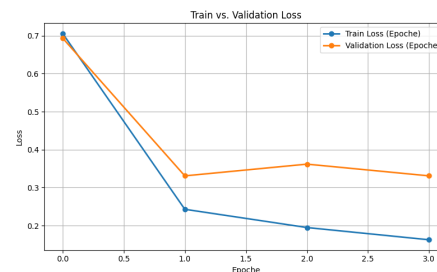


Abbildung 5.4: Train und Validation Loss pro Epoche.

Der Datensatz wurde im Verhältnis von 80 zu 20 in Trainings- und Testmenge aufgeteilt. Das Modell wurde für maximal zehn Epochen trainiert, jedoch griff ein Early-Stopping-Mechanismus, der das Training nach drei Epochen beendete, da sich der $F1$ -Score auf den Validierungsdaten mindestens zwei Epochen lang nicht weiter verbesserte (vgl. Abb. 5.3). Dies deutet darauf hin, dass das Modell ab diesem Zeitpunkt zwar die Trainingsdaten zunehmend besser abbildete, jedoch keine zusätzlichen Generalisierungsgewinne mehr erzielte. Das beobachtete Muster ist typisch für ein beginnendes Overfitting, bei dem die Vorhersagen auf den Validierungsdaten nicht mehr von der stärkeren Anpassung profitieren. Als bestes Modell wurde somit

automatisch der Modellzustand nach der ersten Epoche gespeichert und für die Validierung verwendet. Für das **RoBERTa**-Modell wurde im Tokenizer eine maximale Sequenzlänge von 128 Tokens festgelegt. Diese Begrenzung dient der Reduzierung des Rechenaufwands und der Schonung der verfügbaren Ressourcen während des Trainings und der Inferenz. Zwar überschreiten 37,1 % der Rezensionen diese Länge und werden dadurch gekürzt, jedoch erzielt **RoBERTa** auch mit dieser reduzierten Kontextlänge stabile Ergebnisse (vgl. Kapitel 5.2.2).

Abbildung 5.4 zeigt den Verlauf des Trainings- und Validierungsverlusts über die Epochen. Zu erkennen ist, dass der Trainingsverlust in den ersten 3 Epochen mit einem Abfall von 0,71 auf 0,16 stark abnimmt. Damit zeigt sich, dass das Modell die zugrunde liegenden Muster der Daten erlernt. Der Validierungsverlust fiel zunächst ebenfalls stark von 0,69 in Epoche 0 (sogenannte Baseline Evaluation) auf 0,33 in Epoche 1. Ab der zweiten Epoche stagnierte dieser Wert jedoch und verschlechterte sich teilweise leicht, bevor er in der dritten Epoche wieder auf das Niveau der ersten Epoche zurückging.

5.2.2 Ergebnisse und Evaluation

Klassifikationsgüte

Die Ergebnisse der Klassifikation auf den Testdaten zeigen, dass beide Modelle in der Lage sind, echte und generierte Rezensionen zuverlässig zu unterscheiden. Die logistische Regression erreicht eine Accuracy von 88,3 % bei einem *F1-Score* von 0,88. **RoBERTa** erzielt mit einer Accuracy von 90,1 % und einem *F1-Score* von 0,89 insgesamt leicht bessere Ergebnisse.

Auffällig ist, dass die logistische Regression mit einem *Recall* von 90,3 % einen im Vergleich zu **RoBERTa** (84,7 %) hohen Wert erzielt. Das bedeutet, dass das Modell deutlich besser darin ist, echte Rezensionen nicht fälschlicherweise als „fake“ zu klassifizieren. **RoBERTa** hingegen weist mit einer Präzision von 94,9 % einen höheren Wert auf, was darauf hindeutet, dass es generierte Rezensionen deutlich seltener als echt klassifiziert als die logistische Regression (86,9 %). Die Konfusionsmatrizen in Abbildung 5.5 unterstreichen die unterschiedlichen Schwächen der beiden Modelle. Während die logistische Regression Fake-Rezensionen häufiger übersieht, klassifiziert **RoBERTa** deutlich mehr echte Rezensionen als generiert. Insgesamt schneidet **RoBERTa** jedoch besser ab.

Modell	Accuracy	Precision	Recall	F1-Score
Logistische Regression	0,883	0,869	0,903	0,885
RoBERTa (Epoche 1)	0,901	0,949	0,847	0,896

Tabelle 5.3: Performance-Metriken für beide Klassifikatoren.

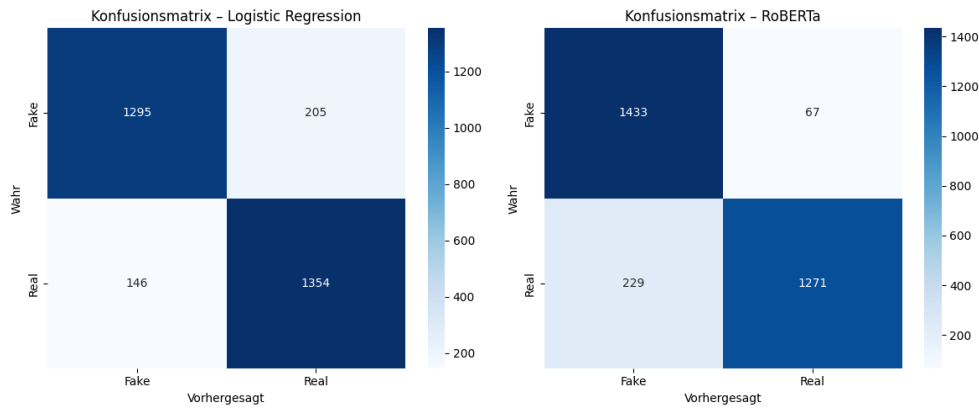


Abbildung 5.5: Konfusionsmatritzen.

Accuracy nach Kategorien

Die Ergebnisse in Tabelle 5.4 zeigen, dass **RoBERTa** in der Mehrheit der Kategorien höhere Genauigkeiten erzielt als die logistische Regression. Dass die Kategorie *Books* durch die logistische Regression besser werden kann, deutet darauf hin, dass die generierten Rezensionen relativ formelhaft und repetitiv gestaltet sind, sodass sie durch einfache Wortfeatures zuverlässig erkannt werden können. Im Gegensatz dazu zeigen die Kategorien *Mobile Apps* und *Music*, dass die generierten Rezensionen hier weniger standardisiert oder floskelhaft sind, sodass sie von einem komplexeren Modell wie **RoBERTa** besser erkannt werden können. Im Anhang A.5 sind jeweils zwei Rezensionen aus der Kategorie *Books*, *Mobile Apps* und *Music* gegenübergestellt, die diese Erkenntnisse unterstützen. Weitere Analysen bestätigen, dass die generierten Rezensionen in den Kategorien *Music* und *Mobile Apps* am häufigsten fälschlicherweise als echt klassifiziert werden. Bei beiden Modellen treten die meisten *False Negatives* in diesen beiden Kategorien auf (vgl. Anhang A.6). Dies deutet darauf hin, dass die generierten Rezensionen hier besonders überzeugend wirken. Gleichzeitig kann diese Beobachtung auch durch die Charakteristika der echten Rezensionen in diesen Kategorien bedingt sein.

Kategorie	Logistische Regression	RoBERTa
Books	0,908	0,878
Mobile Apps	0,866	0,909
Music	0,875	0,925
PC	0,878	0,905
Toys	0,886	0,889

Tabelle 5.4: Kategorieweise Accuracy für **RoBERTa** und logistische Regression.

Insgesamt bleibt die Spannweite der Accuracy bei der logistischen Regression enger (0,866–0,908), während sie bei **RoBERTa** mit Schwankungen zwi-

schen 0,878 und 0,925 etwas stärker variiert. **RoBERTa** ist zwar im Schnitt besser, aber klassifiziert auch weniger stabil über die Kategorien hinweg.

ROC-Kurven

Die Receiver Operating Characteristic (ROC)-Kurven in Abbildung 5.6 veranschaulichen die Trennschärfe der beiden Modelle über verschiedene Schwellenwerte hinweg,

Bei der logistischen Regression verläuft die Kurve oberhalb der Diagonalen und zeigt eine solide Trennfähigkeit. Der berechnete Area Under the Curve (AUC)-Wert liegt bei 0,95, was auf eine zuverlässige, wenn auch nicht perfekte Unterscheidung zwischen echten und generierten Rezensionen hindeutet. Die ROC-Kurve des **RoBERTa**-Modells verläuft minimal näher an der linken oberen Ecke. Mit einem AUC-Wert von 0,97 zeigt **RoBERTa** eine bessere Trennschärfe. Dies bestätigt die aus den Konfusionsmatrizen und Performance-Metriken abgeleitete Überlegenheit des Transformer-Modells.

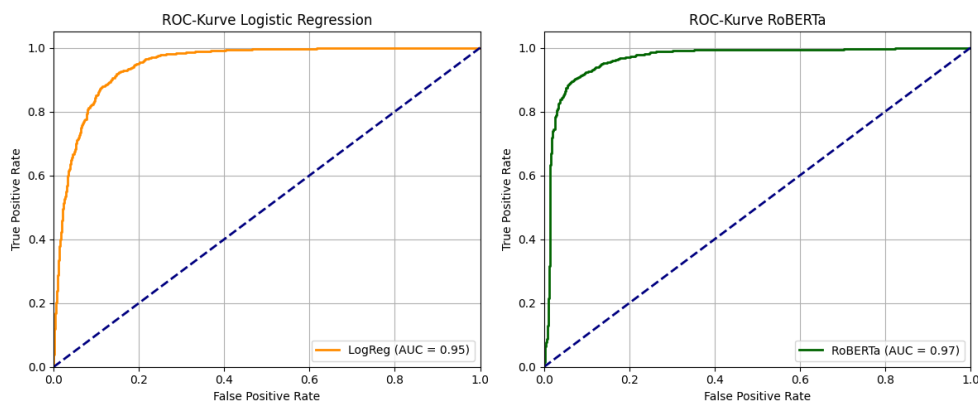


Abbildung 5.6: ROC-Kurven.

Fehlklassifikationen nach Modell

Im Bezug auf die verschiedenen Modelle, die für die Rezensionsgenerierung verwendet wurden, gibt es deutliche Unterschiede in der Performance der Klassifikatoren. Wie in Abbildung 5.5 zu sehen, klassifiziert die logistische Regression ausgewogener falsch. 41,6 % der Fehler betreffen reale Rezensionen und 58,4 % betreffen die generierten Texte. Bei **RoBERTa** entfällt mit 77,4 % aller Fehler ein Großteil der Fehlklassifikationen auf reale Rezensionen. Dies zeigt, dass **RoBERTa** in erster Linie Schwierigkeiten hat, echte Texte korrekt zu erkennen, während generierte Texte vergleichsweise zuverlässig klassifiziert werden.

Bei den generierten Rezensionen fällt insbesondere bei den Texten von *GPT-4.1 Nano (fine-tuned)* eine erhöhte Fehlklassifikationsrate auf, die sich in einer vergleichsweise niedrigen Accuracy widerspiegelt. Sowohl **RoBERTa** (80,5 %) als auch die logistische Regression (47,3 %) haben deutliche Schwierigkeiten, diese Rezensionen zuverlässig von echten zu unterscheiden.

Modell	Fehler LogReg (%)	Accuracy LogReg %	Fehler RoBERTa (%)	Accuracy RoBERTa %
GPT-4.1 Nano	2 (0,6)	99,3	1 (0,3)	99,7
GPT-4.1 Nano (finegetuned)	146 (41,6)	47,3	54 (18,2)	80,5
Llama-3.3-70B-Instruct	13 (3,7)	95,6	2 (0,7)	99,3
Lama-3.1-SauerkrautLM- 70b-Instruct	24 (6,8)	92,2	7 (2,4)	97,7
Gemma-3-27b-It	20 (5,7)	93,7	3 (1,0)	99,1
echte Rezensionen	146 (41,6)	90,3	229 (77,4)	84,7

Tabelle 5.5: Anzahl der Fehlklassifikationen und Accuracy nach Modell für logistische Regression und [RoBERTa](#).

Im Gegensatz dazu schneiden die durch *GPT-4.1 Nano* generierten Rezensionen insgesamt am schlechtesten ab. Dies verdeutlicht, dass das Fine-Tuning einen spürbaren Einfluss auf die Qualität der generierten Rezensionen hatte. Auch bei den [LLaMA](#)-Varianten zeigt die nachtrainierte Version bessere Ergebnisse. Das Modell *Lama-3.1-SauerkrautLM-70b-Instruct*, bei dem 15 % der Layer mit einem deutsch-englischen Datensatz nachtrainiert wurden [59], bereitet beiden Klassifikatoren größere Schwierigkeiten. Gezieltes Nachtrainieren wirkt sich demnach stark auf die Erkennbarkeit und die Qualität der generierten Rezensionen aus. Bereits ein Nachtrainieren auf deutschen Daten führt zu Verbesserungen, während ein zusätzliches Fine-Tuning mit echten Rezensionen die Ergebnisse weiter verbessert.

Zur Untersuchung, ob die beiden Modelle systematisch dieselben Rezensionen falsch klassifizieren, wurden die Fehlklassifikationen von [RoBERTa](#) und der logistischen Regression gegenübergestellt (vgl. Abb. 5.7). Die Venn-Diagramme zeigen dabei die Überlappungen und Unterschiede pro LLM. [RoBERTa](#) und die logistische Regression weisen unterschiedliche Fehlertypen auf und ihre Fehlklassifikationen überlappen nur in geringem Maße. Bei den generierten Rezensionen treten Fehlklassifikationen selten gleichzeitig auf. [RoBERTa](#) zeigt über alle Modelle hinweg eine höhere Robustheit und macht nur vereinzelt (außer bei *GPT-4.1 Nano (finegetuned)*) Fehler bei der Klassifikation. Dies zeigt, dass [RoBERTa](#) stärker kontextuelle Muster und Satzstrukturen nutzt, während die logistische Regression vorrangig auf Wortverteilungen reagiert. Betrachtet man Fehlklassifizierungen und korrekte Klassifizierungen zusammen, treffen die Modelle in 85,1 % der Fälle die gleiche Entscheidung. Die Modelle ergänzen sich damit teilweise in ihren Fehlerbildern. Dies könnte ein Hinweis darauf sein, dass ein Ensemble-Ansatz potenziell die Gesamtperformance weiter verbessern könnte.

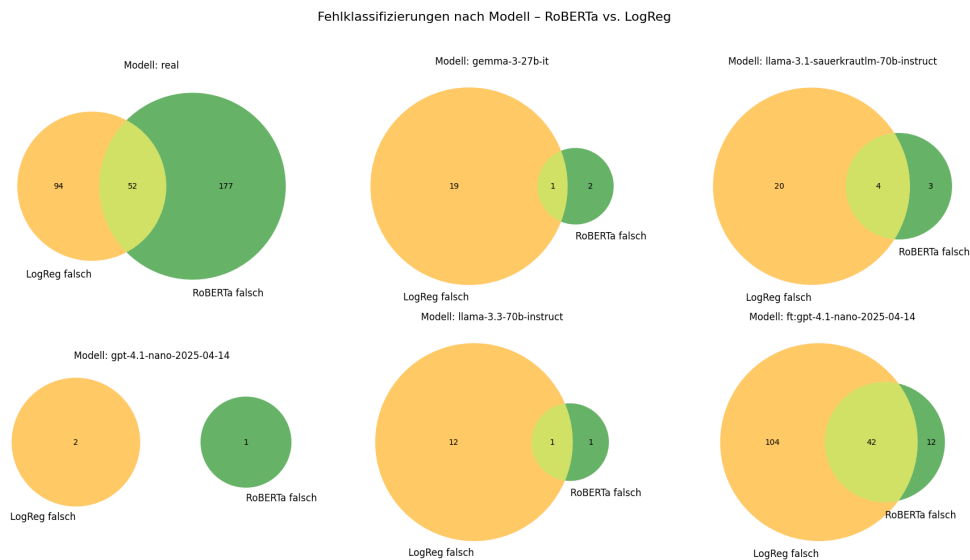


Abbildung 5.7: Venn-Diagramme.

Modellvergleich nach Metadaten

Wortanzahl

Die Analyse der Klassifikationsgenauigkeit in Abhängigkeit von der Textlänge verdeutlicht ebenfalls Unterschiede zwischen den beiden untersuchten Klassifikatoren. Für kurze Rezensionen mit weniger als 50 Wörtern erreichen sowohl **RoBERTa** mit 89,7 % als auch die logistische Regression mit 86,9 % vergleichsweise moderate Genauigkeiten (Abbildung 5.8). Dies lässt sich dadurch erklären, dass kurze Texte in der Regel über einen geringeren semantischen und syntaktischen Informationsgehalt verfügen. Dies erschwert es den Modellen, robuste Unterscheidungsmerkmale zwischen realen und generierten Rezensionen zu extrahieren. Aufgrund der großen Fallzahl (61,6 %) in dieser Kategorie wirkt sich dieser Effekt zudem stark auf die Gesamtgenauigkeit aus.

Im mittleren Längenbereich zwischen 50 und 200 Wörtern steigt die Performanz beider Modelle deutlich an. Besonders für **RoBERTa** erweisen sich die längeren Texte in diesem Segment als vorteilhaft, da die darin enthaltene Vielfalt lexikalischer und syntaktischer Strukturen die Kontextsensitivität [9] transformerbasierter Modelle optimal unterstützt.

Auffällig ist jedoch, dass die logistische Regression bei Texten im Bereich von 150 bis 200 Wörtern mit 93,4 % kurzzeitig bessere Ergebnisse erzielt als **RoBERTa**. Es ist möglich, dass lineare Modelle in klar strukturierten, mittelgroßen Texten von der direkten Gewichtung charakteristischer Schlüsselwörter profitieren können.

Für längere Rezensionen (200–500 Wörter) zeigt sich ein gemischtes Bild. Ab hier sind die Ergebnisse jedoch aufgrund der geringen Stichprobengrößen in den Bins statistisch wenig belastbar. Nur 7,3 % der Rezensionen fallen in diese Bins. So lässt sich auch der Ausreißer nach unten bei den Rezensionen

nen zwischen 300 und 350 Wörtern erklären. Der allgemeine Trend verdeutlicht jedoch, dass sowohl **RoBERTa** als auch die logistischen Rezensionen mit wachsender Rezensionlänge bessere Ergebnisse erzielen.

Trotz der Begrenzung auf 128 Tokens (vgl. Kapitel 5.2.1) zeigt **RoBERTa** insgesamt eine bessere Performance bei längeren Rezensionen als die logistische Regression. Dies legt nahe, dass bereits die ersten 128 Tokens eines Textes in den meisten Fällen genügend semantische und stilistische Informationen enthalten, um eine korrekte Klassifikation vorzunehmen.

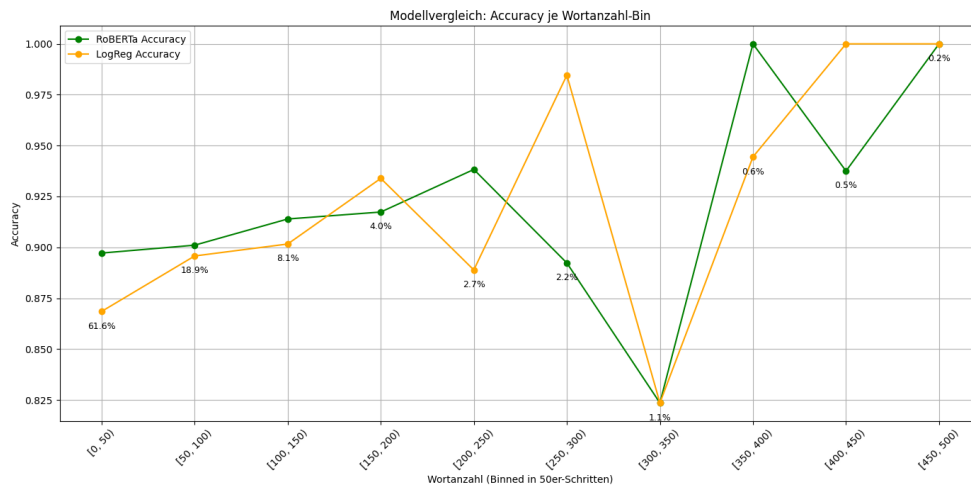


Abbildung 5.8: Accuracy je Wortanzahl-Bin für beide Klassifikatoren: Die Prozentwerte geben den Anteil der Rezensionen für jeden Bin der Test-Daten an.

Rating

Die Untersuchung der Accuracy in Abhängigkeit der Sternebewertungen zeigt, dass beide Modelle bei höheren Ratings tendenziell bessere Leistungen erzielen und sich in gleicher Reihenfolge verbessern. Abbildung 5.9 zeigt auf, dass sowohl **RoBERTa** als auch die logistische Regression für die Vier-Sterne-Rezensionen mit 93,6 % und 91,6 % die höchsten Genauigkeiten erreichen.

Der größte Unterschied in den Genauigkeiten existiert bei der Klassifikation der Ein-Sterne-Rezensionen, bei denen die logistische Regression deutlich schlechter performt. Es lässt darauf schließen, dass lineare Modelle stärker von eindeutigen Schlüsselwörtern und Formulierungen abhängig sind, die in sehr negativen Rezensionen möglicherweise seltener oder variabler auftreten. Diese Annahme wird zusätzlich durch die in Kapitel 5.2.1 vorgestellten *Top 10 Wörter nach Koeffizienten für die logistische Regression* gestützt. Dort zeigt sich, dass Wörter, die besonders stark zur Klassifikation als „fake“ beitragen, überwiegend positiv konnotiert sind.

Auffällig ist außerdem, dass sowohl **RoBERTa** als auch die logistische Regression bei Ein- und Fünf-Sterne-Rezensionen eine niedrigere Genauigkeit erzielen als bei Vier-Sterne-Rezensionen. Ein möglicher Erklärungsansatz dafür liegt in der sprachlichen Differenziertheit von Vier-Sterne-Bewertungen.

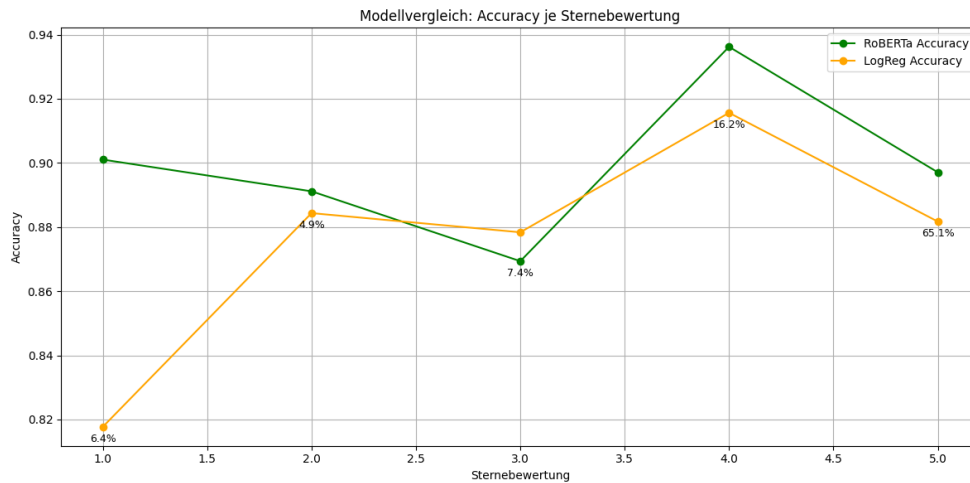


Abbildung 5.9: Accuracy je Sternebewertung für beide Klassifikatoren: Die Prozentwerte geben den Anteil der Rezensionen für jedes Rating der Test-Daten an.

Diese enthalten häufig eine Mischung aus Lob und leichter Kritik, wodurch sprachliche Muster entstehen, die sich besser zur Identifikation generierter Rezensionen eignen.

Demgegenüber weisen Ein- und Fünf-Sterne-Rezensionen oftmals eine hohe Homogenität auf. Sie sind im Durchschnitt kürzer (vgl. Anhang A.7), enthalten standardisierte Floskeln und sind sowohl in realen Rezensionen als auch in generierten Texten stark konventionalisiert. Dadurch verschwimmen die Unterschiede zwischen menschlichen und generierten Rezensionen, was die Klassifikation erschwert.

Zwei- und Drei-Sterne-Rezensionen enthalten häufig authentische, vielfältige weniger floskelhafte Inhalte, die sich sprachlich klar von den eher standardisierten Formulierungen generierter Texte abgrenzen.

FAZIT UND AUSBLICK

Ziel dieser Arbeit war die Erstellung eines qualitativ hochwertigen, gelabelten Datensatzes, der sowohl echte als auch von LLMs generierte Rezensionen umfasst. Mittels Few-Shot-Prompting verschiedener Modelle und gezieltem Fine-Tuning wurden synthetische Rezensionen erzeugt. Beim Few-Shot-Prompting dienten jeweils fünf echte Rezensionen pro Kategorie als Beispiele, die den Modellen als Input bereitgestellt wurden. Die Prompts enthielten zudem strukturierende Informationen wie Wortanzahl, Sternebewertung und Produkttitel, die den Modellen halfen, kohärente und realistische Rezensionen zu erzeugen. Dies ermöglichte die Generierung konsistenter und realitätsnaher Texte, ohne die Modellparameter selbst anzupassen. Das Fine-Tuning beinhaltete eine direkte Anpassung der Modellgewichte auf den spezifischen Stil und die Wortwahl deutscher Rezensionen. Dabei wurden pro Kategorie jeweils 5.000 echte Rezensionen als Trainingsdaten verwendet. Die Auswahl der Trainings- und Prompt-Beispiele aus den echten Rezensionen erfolgte mittels stratifiziertem Sampling, um eine Repräsentativität hinsichtlich Kategorie, Sternebewertung und Wortlänge sicherzustellen.

Die deskriptive Analyse zeigte, dass die Verteilungen der Sternebewertungen und der Textlängen die Originaldaten gut widerspiegeln. Kleinere Abweichungen waren erwartbar, da die Prompts die genaue Wortanzahl nur näherungsweise spezifizierten und die Auswahl zufällig aus stratifizierten Stichproben erfolgte. Insgesamt ist der Datensatz somit statistisch konsistent und für umfangreiche Analysen geeignet. Sprachlich wiesen die generierten Texte eine geringere Diversität auf als die echten Rezensionen. Das Fine-Tuning erhöhte die stilistische Vielfalt und machte die synthetischen Rezensionen schwieriger von menschlich verfassten Texten unterscheidbar.

Im Bezug auf die Datensatzerstellung könnten zukünftig Änderungen im Bezug auf die Prompts (z. B. bezüglich ihrer Länge oder Decoding-Strategie) oder alternative Parameter beim Fine-Tuning verwendet werden, um die Qualität und Variabilität der generierten Rezensionen weiter zu steigern. Aufgrund begrenzter Rechenressourcen war eine Optimierung in größerem Umfang nicht möglich.

Darüber hinaus wurde in dieser Arbeit zur Bewertung der Datensatzqualität lediglich eine einzelne Kennzahl eingesetzt. Für viele Aufgaben der natürlichen Sprachgenerierung ist es sinnvoll, mehrere Metriken heranzuziehen, um verschiedene Aspekte der Textqualität zu erfassen [4]. Darüber hinaus fand im Rahmen dieser Arbeit keine semantische Analyse der generierten Rezensionen statt. Dies stellt eine weitere Einschränkung dar, da die Modelle zwar textliche Muster erkennen, aber nicht zwangsläufig garantieren, dass die generierten Inhalte inhaltlich kohärent oder faktisch plausibel sind. Eine solche semantische Inkonsistenz könnte jedoch ein wichtiges, ungenutztes

Merkmal für zukünftige Klassifikationsmodelle zur Detektion von Opinion Spam darstellen.

Die Klassifikationsversuche mit **RoBERTa** und logistischer Regression ergaben konsistente Ergebnisse. **RoBERTa** erzielte insgesamt die bessere Trennschärfe zwischen echten und generierten Rezensionen. Das Modell tendierte jedoch dazu, reale Rezensionen fälschlich als generiert zu klassifizieren.

Die Modellwahl beeinflussen die Erkennbarkeit deutlich. Das feinjustierte Modell *GPT-4.1 Nano* erzeugte die qualitativ besten und diversesten Rezensionen. *Lama-3.1-SauerkrautLM-70b-Instruct*, das auf deutschsprachigen Daten nachtrainiert wurde, lieferte die zweitbesten Ergebnisse. Im Bezug auf das Fine-Tuning hatten weder die Parameterzahl der **LLMs** noch die Kategorien der Rezensionen einen dominierenden Einfluss auf die Gesamtleistung. Die Klassifikationsqualität variiert jedoch systematisch mit der Textlänge und der Sternebewertung. Sehr kurze Texte waren aufgrund des geringen Informationsgehalts schwieriger zu klassifizieren, mittellange Texte lieferten die zuverlässigsten Ergebnisse. Vier-Sterne-Rezensionen wurden von beiden Modellen am zuverlässigsten erkannt. Dies lässt sich auf die Kombination aus Lob und subtiler Kritik sowie die daraus resultierenden charakteristischen Muster zurückführen. Ein- und Fünf-Sterne-Rezensionen sind aufgrund ihrer Homogenität und Floskelhaftigkeit schwerer zu unterscheiden. Im Hinblick auf die Performanz der eingesetzten Modelle besteht jedoch noch erhebliches Potenzial. Künftige Arbeiten könnten alternative Klassifikatoren oder Ensemble-Methoden einbeziehen, um Fehlerüberlappungen systematisch zu reduzieren. Auch die maximale Tokenlänge des **RoBERTa**-Klassifikators könnte erhöht werden, um sicherzustellen, dass der gesamte Inhalt aller Rezensionen beim Training berücksichtigt und analysiert werden kann.

Gleichzeitig weist die Arbeit auch Limitationen auf. Im weiteren Kontext des Feldes Opinion Spam muss berücksichtigt werden, dass gefälschte Rezensionen auf Plattformen in unterschiedlicher Form auftreten. So existieren beispielsweise Fälle, in denen Käufer*innen Produkte tatsächlich erwerben, diese positiv bewerten und anschließend von den Unternehmen eine Rückerstattung erhalten. Solche Rezensionen sind formal verifiziert, stellen jedoch dennoch eine Form manipulierter Inhalte dar. Ein regelrechter Markt dafür hat sich insbesondere im Umfeld von Amazon entwickelt [20].

Trotz dieser Einschränkungen schließt der hier erstellte Datensatz, im Vergleich zu bestehenden Arbeiten, eine wichtige Forschungslücke zur Opinion-Spam-Erkennung, insbesondere im deutschsprachigen Raum. Während bisherige Datensätze oft nur wenige hundert bis tausend Rezensionen umfassen oder nicht öffentlich zugänglich sind, stellt dieser Datensatz eine qualitativ hochwertige Grundlage dar, der die weiterführende Forschung zur Erkennung und Charakterisierung gefälschter Rezensionen ermöglicht.

Teil II

APPENDIX



ANHANG

A.1 BEISPIEL-PROMPT

```
[
  {
    "role": "system",
    "content": "Du bist ein Käufer oder eine Käuferin, der/die ehrliche,
      natürlich klingende Amazon-Produktrezensionen schreibt.
      Vermeide unnötige Wiederholungen.
      Schreibe flüssig und in Alltagssprache.
      Halte dich an den angegebenen Wortumfang.
      Keine formelle Sprache - schreibe so, wie echte Kund:innen schreiben
      würden."
  },
  {
    "role": "user",
    "content": "Produkt: Janod 4506520 - Maxi Küche Chic, 8 Accessoires
      Bewertung: 1"
  },
  {
    "role": "assistant",
    "content": "Unsere Tochter sollte eine Holz-Kinderküche zu
      Weihnachten bekommen. ... ansonsten können wir nur vom Kauf
      abraten."
  },
  {
    "role": "user",
    "content": "Produkt: Lego Star Wars 7913 - Clone Trooper Battle Pack
      Bewertung: 2"
  },
  {
    "role": "assistant",
    "content": "Mein Sohn freute sich sehr über das Teil, jedoch mussten
      wir feststellen, dass 4 Klein-Teile fehlten. ... Nicht gut!"
  },
  {
    "role": "user",
    "content": "Produkt: Lego Star Wars 9494 - Anakins Jedi Interceptor
      Bewertung: 3"
  },
  {
    "role": "assistant",
    "content": "Leider zählt man hier den Star-Wars-Zuschlag zu sehr:
      wenig Teile für viel Geld."
  },
  {
```

```
"role": "user",
"content": "Produkt: Lego Technic 42023 - Baustellen-Set
Bewertung: 4"
},
{
"role": "assistant",
"content": "3 Baustellenfahrzeuge in einem Set ist einfach nur
Spitze. Freue mich schon auf die großen Augen meines Enkels zu
Weihnachten."
},
{
"role": "user",
"content": "Produkt: AMIGO 07900 - Bohnanza, Kartenspiel
Bewertung: 5"
},
{
"role": "assistant",
"content": "Ein Spiel, das ohne viel Material spannend und
abwechslungsreich ist. Selbst nach mehreren Runden bleibt der
Spaß erhalten. Glück und Taktik sind gut ausbalanciert."
},
{
"role": "user",
"content": "Produkt: Revell 06723 - easykit Steckbausatz Star Wars,
X-wing Fighter
Bewertung: 3
Schreibe ungefähr 26 Wörter."
}
]
```

Listing A.1: vollständiger Few-Shot-Prompt für die Rezensionsgenierung der Kategorie *Toys*.

A.2 REZENSION FÜR FINE-TUNING

```
[
  {
    "role": "system",
    "content": "Du bist ein Käufer oder eine Käuferin, der/die
      ehrliche, natürlich klingende Amazon-Produktrezensionen
      schreibt.
      Vermeide unnötige Wiederholungen.
      Schreibe flüssig und in Alltagssprache.
      Halte dich an den angegebenen Wortumfang.
      Keine formelle Sprache - schreibe so, wie echte Kund:innen
      schreiben würden."
  },
  {
    "role": "user",
    "content": "Produkt: Brio 33214 - Batterie-Frachtlok
      Bewertung: 1
      Schreibe ungefähr 26 Wörter."
  },
  {
    "role": "assistant",
    "content": "Viele Worte sind nicht nötig. Das Licht funktioniert
      noch, sonst nichts. Da dies kein Einzelfall ist, muss ich
      den Verkauf dieser Lok generell als Frechheit bezeichnen."
  }
]
```

Listing A.2: Rezension im JSON-Format für Fine-Tuning.

A.3 ORIGINALVERLAUF OHNE GLÄTTUNG DES FINE-TUNINGS

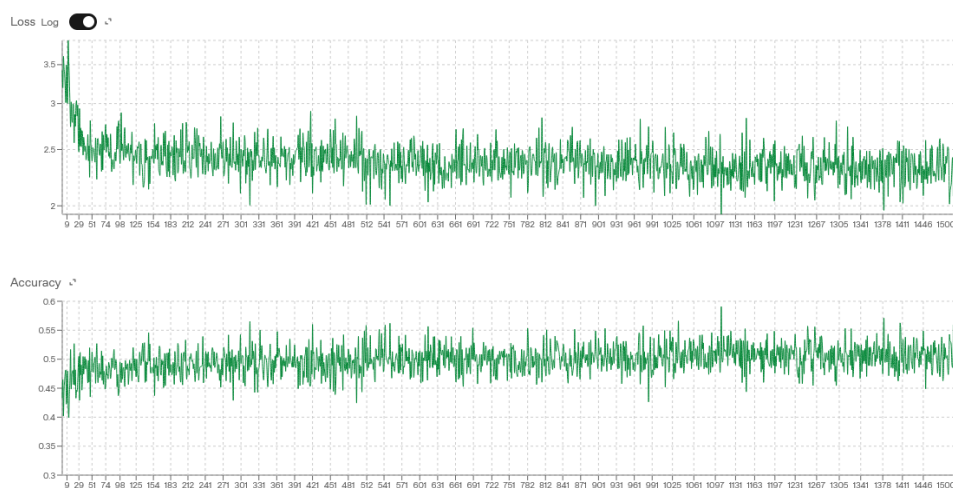
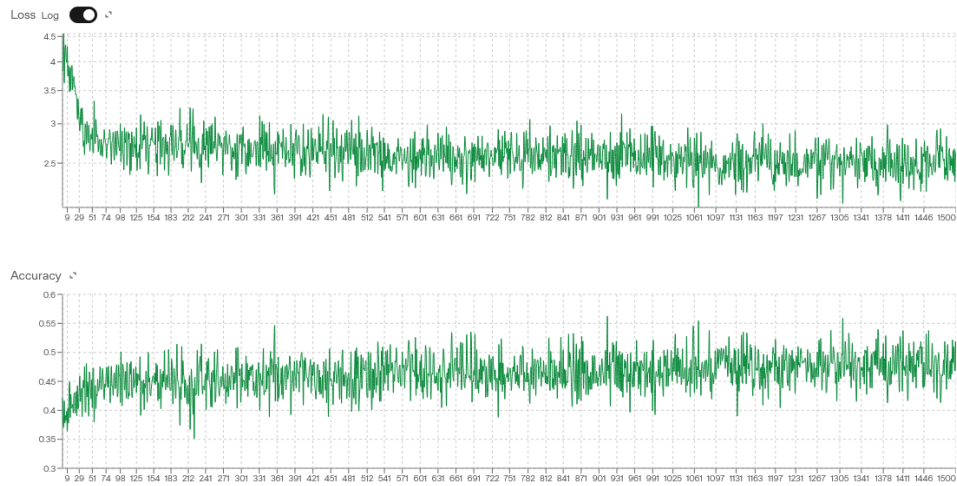
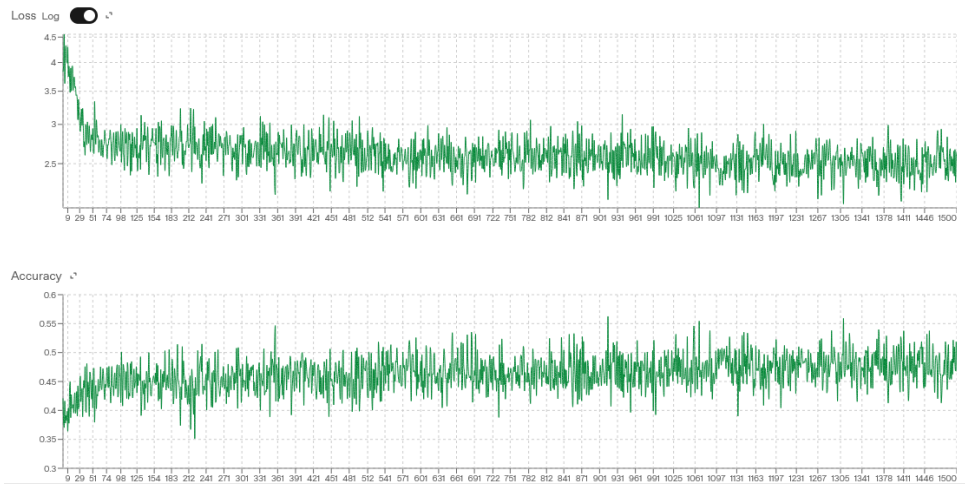
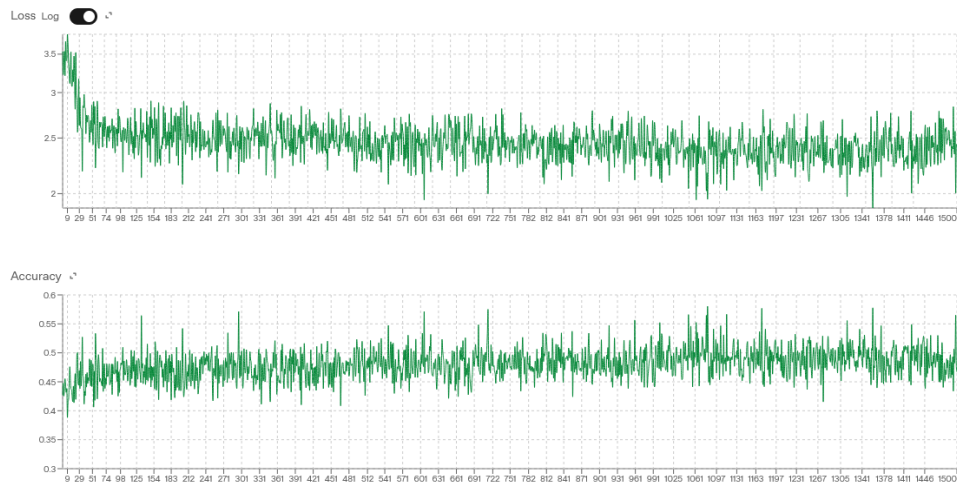
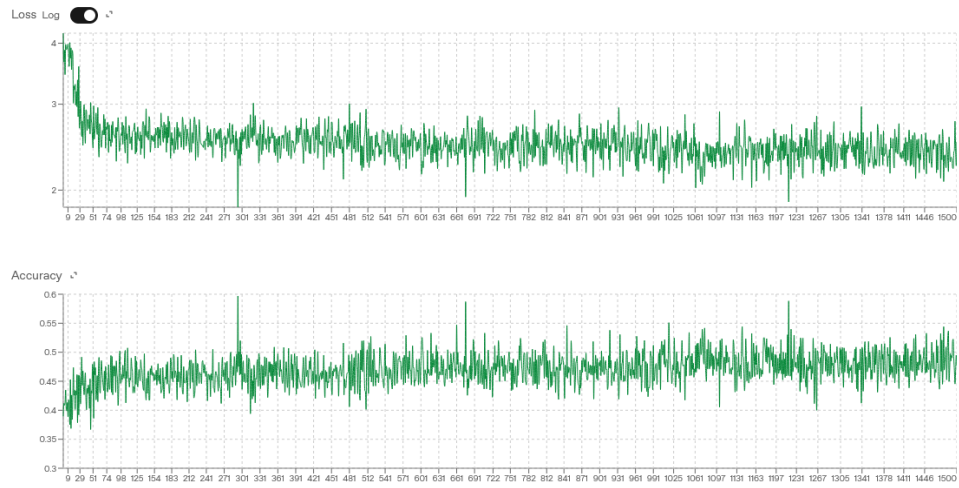


Abbildung A.1: Loss und Accuracy Fine-Tuning Books.

Abbildung A.2: Loss und Accuracy Fine-Tuning *Mobile Apps*.Abbildung A.3: Loss und Accuracy Fine-Tuning *Music*.Abbildung A.4: Loss und Accuracy Fine-Tuning *PC*.

Abbildung A.5: Loss und Accuracy Fine-Tuning *Toys*.

A.4 BEISPIELREZENSIONEN PRO MODELL

Modell	Beispielrezension
GPT-4.1 Nano	„Das Album ist okay, aber nichts wirklich Neues. Manche Songs gefallen mir, andere sind eher durchschnittlich. Für Fans sicher ganz in Ordnung, mehr aber auch nicht.“
GPT-4.1 Nano (fingetuned)	„U2 hat den „alten“ Sound wieder entdeckt und mit ein wenig neuen Sounds gemixt. Leider keine CD für's Auto, sondern nur was für zu Hause.“
Llama-3.1-SauerkrautLM-70b-Instruct	„Schade, dass es nicht mehr so wie die alten Meisterwerke geworden ist. Noch ok, aber kein Highlight.“
Llama-3.3-70B-Instruct	„Gutes Album, aber nicht überragend. Einige Tracks sind sehr schön, andere eher durchschnittlich. Für Fans von Michael Jackson sicherlich eine Empfehlung, aber nicht sein Meisterwerk.“
Gemma-3-27b-It	„Interessantes Konzeptalbum, aber nicht ganz meins. Die Kombination aus elektronischen Klängen und klassischen Instrumenten ist gewöhnungsbedürftig. Einige Tracks gefallen mir gut, andere wirken etwas zu experimentell und gehen unter. Insgesamt solide, aber kein Highlight.“
real	„War für mich die Bestätigung, dass sich der Musikgeschmack mit dem Alter wirklich verändert. Hatte die Scheibe viel besser in Erinnerung. Höre die neueren Platten aber immer noch gerne...“

Tabelle A.1: 3-Sterne-Beispielrezensionen der Kategorie *Music* aus dem Datensatz.

A.5 GENERIERTE REZENSIONEN DER KATEGORIEN BOOKS, MOBILE APPS UND MUSIC

Kategorie	Rezension
<i>Books</i>	<p>Beispiel 1: „Dieses Buch ist eines der besten die ich je gelesen habe. Es hat mich sehr berührt und zum Nachdenken gebracht.“</p> <p>Beispiel 2: „Dieses Buch ist einfach atemberaubend. Es lässt mich meine eigene Sicht der Dinge überdenken und hat mir neue Perspektiven eröffnet.“</p>
<i>Mobile Apps</i>	<p>Beispiel 1: „Leider funktioniert die App nur schlecht auf meinem Kindle Fire HD 8. Die Filme beginnen zu ruckeln und brechen oft ab.“</p> <p>Beispiel 2: „Der Spielstand ist ständig weg und die Anzeige von Werbung nervt. Ich muss leider sagen, dass ich enttäuscht bin von dem Spiel.“</p>
<i>Music</i>	<p>Beispiel 1: „Einzigartige Soundtrack-Kombination, die perfekt zum Film passt, mit Western-Klassikern und modernen Elementen.“</p> <p>Beispiel 2: „David Foster ist einfach ein Genie. Diese CD ist ein Must-Have für jeden David-Foster-Fan, aber auch für Freunde von Qualitätspop!“</p>

Tabelle A.2: Generierte Rezensionen der Kategorie *Books*, *Mobile Apps* und *Music* aus dem Datensatz.

A.6 FALSE NEGATIVES

Produktkategorie	RoBERTa (FN)	Logistische Regression (FN)
Mobile Apps	17	48
Music	15	46
Books	12	38
Toys	12	38
PC	11	35

Tabelle A.3: Anzahl der *False Negatives* pro Modell und Produktkategorie.

A.7 REZENSIONSLÄNGE JE STERNEBewERTUNG

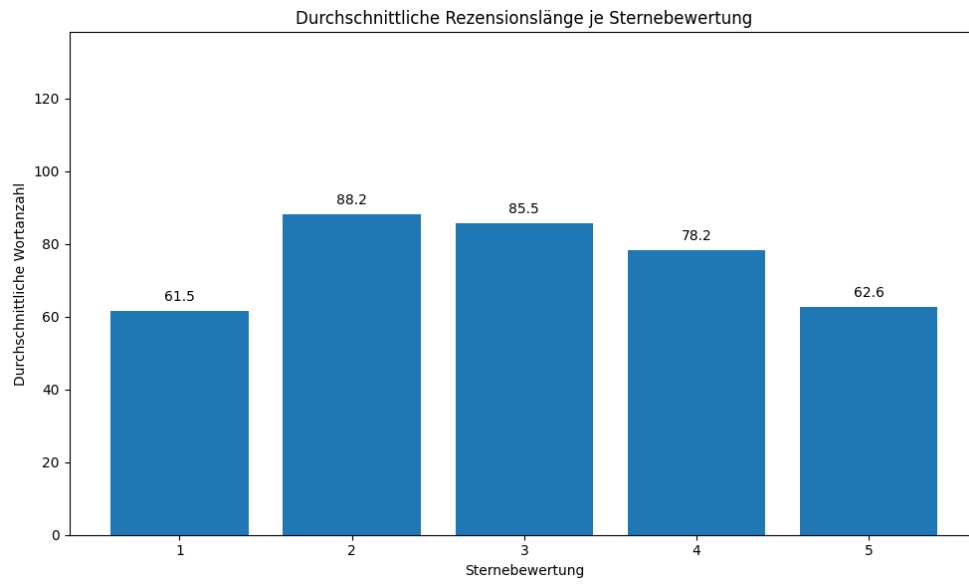


Abbildung A.6: durchschnittliche Rezensionslänge je Sternebewertung.

LITERATUR

- [1] Iz Beltagy, Matthew E. Peters und Arman Cohan. *Longformer: The Long-Document Transformer*. 2020. URL: <https://arxiv.org/abs/2004.05150>.
- [2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent und Christian Janvin. "A neural probabilistic language model". In: *J. Mach. Learn. Res.* 3.null (März 2003), 1137–1155. ISSN: 1532-4435.
- [3] Tom B. Brown et al. "Language Models are Few-Shot Learners". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS '20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546. URL: <https://arxiv.org/abs/2005.14165>.
- [4] Asli Celikyilmaz, Elizabeth Clark und Jianfeng Gao. *Evaluation of Text Generation: A Survey*. 2021. URL: <https://arxiv.org/abs/2006.14799>.
- [5] Davide Chicco und Giuseppe Jurman. "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". In: *BMC Genomics* 21.1 (2020), S. 6. DOI: [10.1186/s12864-019-6413-7](https://doi.org/10.1186/s12864-019-6413-7).
- [6] Hyung Won Chung et al. "Scaling instruction-finetuned language models". In: *J. Mach. Learn. Res.* 25.1 (Jan. 2024). ISSN: 1532-4435.
- [7] Jacopo D'Abramo, Andrea Zugarini und Paolo Torrioni. *Dynamic Few-Shot Learning for Knowledge Graph Question Answering*. 2024. URL: <https://arxiv.org/abs/2407.01409>.
- [8] Michal Mimino Danilak. *langdetect*. Version 1.0.9, veröffentlicht am 7. Mai 2021; Zugriff am 14. November 2025. 2021. URL: <https://pypi.org/project/langdetect/>.
- [9] Jacob Devlin, Ming-Wei Chang, Kenton Lee und Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *North American Chapter of the Association for Computational Linguistics*. 2019. URL: <https://api.semanticscholar.org/CorpusID:52967399>.
- [10] Yogesh K. Dwivedi et al. "Setting the future of digital and social media marketing research: Perspectives and research propositions". In: *International Journal of Information Management* 59 (2021), S. 102168. ISSN: 0268-4012. DOI: <https://doi.org/10.1016/j.ijinfomgt.2020.102168>.
- [11] Ludwig Fahrmeir, Rita Künstler, Iris Pigeot und Gerhard Tutz. *Statistik: Der Weg zur Datenanalyse*. 6. Aufl. Springer-Lehrbuch. Springer Berlin Heidelberg, 2007. ISBN: 978-3-540-69739-8. DOI: [10.1007/978-3-540-69739-8](https://doi.org/10.1007/978-3-540-69739-8).

- [12] Angela Fan, Mike Lewis und Yann Dauphin. “Hierarchical Neural Story Generation”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Hrsg. von Iryna Gurevych und Yusuke Miyao. Melbourne, Australia: Association for Computational Linguistics, Juli 2018, S. 889–898. DOI: [10.18653/v1/P18-1082](https://doi.org/10.18653/v1/P18-1082).
- [13] Donato Hernández Fusilier, Manuel Montes-y Gómez, Paolo Rosso und Rafael Guzmán Cabrera. “Detection of Opinion Spam with Character n-grams”. In: *Computational Linguistics and Intelligent Text Processing*. Hrsg. von Alexander Gelbukh. Springer International Publishing, 2015, S. 285–294. ISBN: 978-3-319-18117-2.
- [14] N. Gobi und A. Rathinavelu. “Analyzing cloud based reviews for product ranking using feature based clustering algorithm”. In: *Cluster Computing* 22.Suppl 3 (2019), S. 6977–6984. DOI: [10.1007/s10586-018-1996-3](https://doi.org/10.1007/s10586-018-1996-3).
- [15] Aaron Grattafiori et al. *The Llama 3 Herd of Models*. 2024. URL: <https://arxiv.org/abs/2407.21783>.
- [16] Eric Hartford, Lucas Atkins, Fernando Fernandes Neto und David Golchinfar. *Spectrum: Targeted Training on Signal to Noise Ratio*. 2024. URL: <https://arxiv.org/abs/2406.06623>.
- [17] Trevor Hastie, Robert Tibshirani und Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.
- [18] Kaiming He, Xiangyu Zhang, Shaoqing Ren und Jian Sun. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, S. 770–778. DOI: [10.1109/CVPR.2016.90](https://doi.org/10.1109/CVPR.2016.90).
- [19] Sherry He, Brett Hollenbeck und Davide Proserpio. “The Market for Fake Reviews”. In: *Marketing Science* 41.5 (2022). Available at SSRN, S. 896–921. DOI: [10.2139/ssrn.3664992](https://doi.org/10.2139/ssrn.3664992).
- [20] Simon Hill. *Inside the Underground Market for Fake Amazon Reviews*. Zugriff am 14. November 2025. 2022. URL: <https://www.wired.com/story/fake-amazon-reviews-underground-market/>.
- [21] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes und Yejin Choi. “The Curious Case of Neural Text Degeneration”. In: *International Conference on Learning Representations*. 2020. URL: <https://openreview.net/forum?id=rygGQyrFvH>.
- [22] Edward J Hu, Yelong Yhen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang und Weizhu Chen. “LoRA: Low-Rank Adaptation of Large Language Models”. In: *International Conference on Learning Representations*. 2022. URL: <https://openreview.net/forum?id=nZeVKeeFYf9>.

- [23] Nitin Jindal und Bing Liu. "Opinion spam and analysis". In: *Proceedings of the 2008 International Conference on Web Search and Data Mining*. WSDM '08. Palo Alto, California, USA: Association for Computing Machinery, 2008, 219–230. ISBN: 9781595939272. DOI: [10.1145/1341531.1341560](https://doi.org/10.1145/1341531.1341560).
- [24] Daniel Jurafsky und James Martin. *Speech and Language Processing Pearson New International Edition*. Pearson Deutschland, 2013. ISBN: 9781292025438. URL: <https://elibrary.pearson.de/book/99.150005/9781292037936>.
- [25] Feyza Duman Keles, Pruthuvi Maheshakya Wijewardena und Chinmay Hegde. "On The Computational Complexity of Self-Attention". In: *International Conference on Algorithmic Learning Theory*. 2022. URL: <https://api.semanticscholar.org/CorpusID:252198880>.
- [26] Phillip Keung, Yichao Lu, György Szarvas und Noah A. Smith. *The Multilingual Amazon Reviews Corpus*. 2020. arXiv: [2010.02573](https://arxiv.org/abs/2010.02573) [cs.CL]. URL: <https://arxiv.org/abs/2010.02573>.
- [27] P. Koehn. *Statistical Machine Translation*. Statistical Machine Translation. Cambridge University Press, 2010. ISBN: 9780521874151. URL: https://books.google.de/books?id=4v_Cx1wIMLkC.
- [28] Jianing Li, Yanyan Lan, Jiafeng Guo und Xueqi Cheng. "On the Relation between Quality-Diversity Evaluation and Distribution-Fitting Goal in Text Generation". In: *Proceedings of the 37th International Conference on Machine Learning*. Hrsg. von Hal Daumé III und Aarti Singh. Bd. 119. Proceedings of Machine Learning Research. PMLR, 2020, S. 5905–5915. URL: <https://proceedings.mlr.press/v119/li20h.html>.
- [29] Bing Liu. "Sentiment Analysis and Subjectivity". In: *Handbook of Natural Language Processing*. Chapman und Hall/CRC, 2010, S. 627–666. ISBN: 978-1-4200-8593-8. URL: <http://dblp.uni-trier.de/db/referen ce/nlp/nlp2010.html#Liu10>.
- [30] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer und Veselin Stoyanov. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. URL: <https://arxiv.org/abs/1907.11692>.
- [31] Vijini Liyanage, Davide Buscaldi und Penelope Forcioli. "Detecting AI-enhanced Opinion Spambots: a study on LLM-generated Hotel Reviews". In: *Proceedings of the Seventh Workshop on e-Commerce and NLP @ LREC-COLING 2024*. Hrsg. von Shervin Malmasi, Besnik Fetahu, Nicola Ueffing, Oleg Rokhlenko, Eugene Agichtein und Ido Guy. Torino, Italia: ELRA und ICCL, Mai 2024, S. 74–78. URL: <https://aclanthology.org/2024.ecnlp-1.8/>.
- [32] Michael Luca. *Reviews, Reputation, and Revenue: The Case of Yelp.Com*. NOM Unit Working Paper 12-016. Harvard Business School, 2016. DOI: [10.2139/ssrn.1928601](https://doi.org/10.2139/ssrn.1928601).

- [33] Thang Luong, Hieu Pham und Christopher D. Manning. “Effective Approaches to Attention-based Neural Machine Translation”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Hrsg. von Lluís Màrquez, Chris Callison-Burch und Jian Su. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, S. 1412–1421. DOI: [10.18653/v1/D15-1166](https://doi.org/10.18653/v1/D15-1166).
- [34] Ggaliwango Marvin, Nakayiza Hellen, Daudi Jjingo und Joyce Nakatumba-Nabende. “Prompt Engineering in Large Language Models”. In: *Data Intelligence and Cognitive Informatics*. Hrsg. von I. Jeena Jacob, Selwyn Piramuthu und Przemyslaw Falkowski-Gilski. Singapore: Springer Nature Singapore, 2024. ISBN: 978-981-99-7962-2. DOI: https://doi.org/10.1007/978-981-99-7962-2_30.
- [35] Meta AI. *Introducing Meta Llama 3: The most capable openly available LLM to date*. Meta AI Blog. Zugriff am 10. Dezember 2025. 2024. URL: <https://ai.meta.com/blog/meta-llama-3/>.
- [36] Microsoft. *Customize a model with fine-tuning*. Microsoft Learn. Zugriff am 10. Dezember 2025. 2025. URL: <https://learn.microsoft.com/en-us/azure/ai-foundry/openai/how-to/fine-tuning>.
- [37] Arjun Mukherjee. “Detecting Deceptive Opinion Spam using Linguistics, Behavioral and Statistical Modeling”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing: Tutorial Abstracts*. Hrsg. von Eneko Agirre und Kevin Duh. Beijing, China: Association for Computational Linguistics, Juli 2015, S. 21–22. DOI: [10.3115/v1/P15-5007](https://doi.org/10.3115/v1/P15-5007).
- [38] Rohit Narayan, Jitendra Kumar Rout und Sanjay Kumar Jena. “Review Spam Detection Using Opinion Mining”. In: *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*. Hrsg. von Pan-kaj Kumar Sa, Manmath Narayan Sahoo, M. Murugappan, Yulei Wu und Banshidhar Majhi. Singapore: Springer Singapore, 2018, S. 273–279. ISBN: 978-981-10-3376-6. URL: https://link.springer.com/chapter/10.1007/978-981-10-3376-6_30.
- [39] OpenAI. *Chat Completions | OpenAI API Reference*. OpenAI Platform Documentation. Zugriff am 11. Dezember 2025. URL: <https://platform.openai.com/docs/api-reference/chat/create>.
- [40] OpenAI. *Fine-tuning | OpenAI API Reference*. OpenAI Platform Documentation. Zugriff am 12. Dezember 2025. URL: <https://platform.openai.com/docs/api-reference/fine-tuning>.
- [41] OpenAI. *Supervised fine-tuning | OpenAI API*. OpenAI Platform Documentation. Zugriff am 12. Dezember 2025. URL: <https://platform.openai.com/docs/guides/supervised-fine-tuning>.
- [42] OpenAI. *Introducing GPT-4.1 in the API*. OpenAI API Documentation. Zugriff am 12. Dezember 2025. 2025. URL: <https://openai.com/index/gpt-4-1/>.

- [43] OpenAI et al. *GPT-4 Technical Report*. 2024. URL: <https://arxiv.org/abs/2303.08774>.
- [44] Myle Ott, Yejin Choi, Claire Cardie und Jeffrey T. Hancock. "Finding Deceptive Opinion Spam by Any Stretch of the Imagination". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Hrsg. von Dekang Lin, Yuji Matsumoto und Rada Mihalcea. Portland, Oregon, USA: Association for Computational Linguistics, Juni 2011, S. 309–319. URL: <https://aclanthology.org/P11-1032/>.
- [45] Kishore Papineni, Salim Roukos, Todd Ward und Wei-Jing Zhu. "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Hrsg. von Pierre Isabelle, Eugene Charniak und Dekang Lin. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Juli 2002, S. 311–318. DOI: [10.3115/1073083.1073135](https://doi.org/10.3115/1073083.1073135).
- [46] Fritz Pokropp. "Kapitel 3. Geschichtete Stichproben (Stratified Sampling)". In: *Stichproben – Theorie und Verfahren*. Berlin, Boston: Oldenbourg Wissenschaftsverlag, 1996, S. 51–74. ISBN: 9783486791303. DOI: [doi:10.1515/9783486791303-004](https://doi.org/10.1515/9783486791303-004).
- [47] Atika Qazi, Najmul Hasan, Rui Mao, Mohamed Elhag Mohamed Abo, Samrat Kumar Dey und Glenn Hardaker. "Machine Learning-Based Opinion Spam Detection: A Systematic Literature Review". In: *IEEE Access* 12 (2024), S. 143485–143499. DOI: [10.1109/ACCESS.2024.3399264](https://doi.org/10.1109/ACCESS.2024.3399264).
- [48] Alec Radford, Karthik Narasimhan, Tim Salimans und Ilya Sutskever. *Improving Language Understanding by Generative Pre-Training*. OpenAI Blog. Zugriff am 10. Dezember 2025. 2018. URL: <https://openai.com/index/language-unsupervised/>.
- [49] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li und Peter J. Liu. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140 (2020), S. 1–67. URL: <http://jmlr.org/papers/v21/20-074.html>.
- [50] Ajay Rastogi und Monica Mehrotra. "Impact of Behavioral and Textual Features on Opinion Spam Detection". In: *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*. 2018, S. 852–857. DOI: [10.1109/ICCONS.2018.8662912](https://doi.org/10.1109/ICCONS.2018.8662912).
- [51] *Richtlinie (EU) 2019/2161 des Europäischen Parlaments und des Rates vom 27. November 2019 zur Änderung der Richtlinie 93/13/EWG des Rates und der Richtlinien 98/6/EG, 2005/29/EG und 2011/83/EU zur besseren Durchsetzung und Modernisierung der Verbraucherschutzvorschriften der Union*. Amtsblatt der Europäischen Union, L 328, 18.12.2019, S. 7–28. 2019. URL: <https://eur-lex.europa.eu/legal-content/DE/TXT/?uri=CELEX:32019L2161>.

- [52] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal und Aman Chadha. *A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications*. 2025. arXiv: [2402.07927](https://arxiv.org/abs/2402.07927).
- [53] Joni Salminen, Chandrashekar Kandpal, Ahmed Mohamed Kamel, Soon gyo Jung und Bernard J. Jansen. "Creating and detecting fake reviews of online products". In: *Journal of Retailing and Consumer Services* 64 (2022), S. 102771. ISSN: 0969-6989. DOI: <https://doi.org/10.1016/j.jretconser.2021.102771>.
- [54] "TF-IDF". In: *Encyclopedia of Machine Learning*. Hrsg. von Claude Sammut und Geoffrey I. Webb. Springer US, 2010, S. 986–987. ISBN: 978-0-387-30164-8. DOI: [10.1007/978-0-387-30164-8_832](https://doi.org/10.1007/978-0-387-30164-8_832).
- [55] Vlad Sandulescu und Martin Ester. "Detecting Singleton Review Spammers Using Semantic Similarity". In: *Proceedings of the 24th International Conference on World Wide Web*. WWW '15. ACM, Mai 2015, 971–976. DOI: [10.1145/2740908.2742570](https://doi.org/10.1145/2740908.2742570).
- [56] Melanie Siegel. "Opinion Spam – Meinungsäußerungen als Fake". In: *Information - Wissenschaft Praxis* 67.5-6 (2016), S. 304–310. DOI: [doi:10.1515/iwp-2016-0058](https://doi.org/10.1515/iwp-2016-0058).
- [57] Aaron Smith und Monica Anderson. *Online Shopping and E-Commerce: Online Reviews*. Pew Research Center. 2016. URL: <https://www.pewinternet.org/2016/12/19/online-reviews/>.
- [58] Marina Sokolova und Guy Lapalme. "A systematic analysis of performance measures for classification tasks". In: *Information Processing Management* 45.4 (2009), S. 427–437. ISSN: 0306-4573. DOI: <https://doi.org/10.1016/j.ipm.2009.03.002>.
- [59] VAGO Solutions. *Llama-3.1-SauerkrautLM-70b-Instruct*. Zugriff am 12. Dezember 2025, 2024. URL: <https://huggingface.co/VAGOsolutions/Llama-3.1-SauerkrautLM-70b-Instruct>.
- [60] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo und Yunfeng Liu. "RoFormer: Enhanced transformer with Rotary Position Embedding". In: *Neurocomputing* 568 (2024), S. 127063. ISSN: 0925-2312. DOI: <https://doi.org/10.1016/j.neucom.2023.127063>.
- [61] Gemma Team et al. *Gemma: Open Models Based on Gemini Research and Technology*. 2024. arXiv: [2403.08295](https://arxiv.org/abs/2403.08295) [cs.CL].
- [62] Gemma Team et al. *Gemma 3 Technical Report*. 2025. arXiv: [2503.19786](https://arxiv.org/abs/2503.19786) [cs.CL].
- [63] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: [2302.13971](https://arxiv.org/abs/2302.13971) [cs.CL].
- [64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser und Illia Polosukhin. *Attention Is All You Need*. 2023. arXiv: [1706.03762](https://arxiv.org/abs/1706.03762) [cs.CL]. URL: <https://arxiv.org/abs/1706.03762>.

- [65] *Verordnung (EU) 2022/2065 des Europäischen Parlaments und des Rates vom 19. Oktober 2022 über einen Binnenmarkt für digitale Dienste und zur Änderung der Richtlinie 2000/31/EG (Gesetz über digitale Dienste)*. Amtsblatt der Europäischen Union, L 277, 27.10.2022, S. 1–102. 2022. URL: <https://eur-lex.europa.eu/legal-content/DE/TXT/?uri=CELEX:32022R2065>.
- [66] Lianguo Wang, Ruoyu Chen und Li Li. “Knowledge-Guided Prompt Learning for Few-Shot Text Classification”. In: *Electronics* 12.6 (2023). ISSN: 2079-9292. DOI: [10.3390/electronics12061486](https://doi.org/10.3390/electronics12061486).
- [67] Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho und Jason Weston. “Neural Text Generation With Unlikelihood Training”. In: *International Conference on Learning Representations*. 2020. URL: <https://openreview.net/forum?id=SJeYe0NtvH>.
- [68] Xiao-Kun Wu et al. “LLM Fine-Tuning: Concepts, Opportunities, and Challenges”. In: *Big Data and Cognitive Computing* 9.4 (2025). ISSN: 2504-2289. URL: <https://www.mdpi.com/2504-2289/9/4/87>.
- [69] Runcheng Xia, Xuefan Dong, Jianjun An und Handi Wang. “The impact of fake online reviews on customer satisfaction: an empirical study on JD.com”. In: *Electronic Commerce Research* (2024). DOI: [10.1007/s10660-024-09865-y](https://doi.org/10.1007/s10660-024-09865-y).
- [70] Biao Zhang und Rico Sennrich. “Root Mean Square Layer Normalization”. In: *Advances in Neural Information Processing Systems*. Hrsg. von H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox und R. Garnett. Bd. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf.
- [71] Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang und Yong Yu. “Texygen: A Benchmarking Platform for Text Generation Models”. In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '18. Ann Arbor, MI, USA: Association for Computing Machinery, 2018, 1097–1100. ISBN: 9781450356572. DOI: [10.1145/3209978.3210080](https://doi.org/10.1145/3209978.3210080).
- [72] scikit-learn developers. *LogisticRegression - scikit learn 1.8.0 documentation*. Zugriff am 28. Dezember 2025. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.