Department of Data Science & Artificial Intelligence

Master Thesis

Robuste Erkennung von KI-generierten Texten in deutscher Sprache

Abgabedatum:

30. August 2023

Eingereicht von:
Tom Tlok
Am Domplatz 6
21220 Seevetal
Tel.: 0160 - 94694134
E-Mail: dsai105669@stud.fh-wedel.de
Verwaltungssemester: 4. Semester

Referent:
Prof. Dr. Hendrik Annuth
Fachhochschule Wedel
Feldstraße 143
22880 Wedel
Telefon: 04103 - 8048 - 18
E-Mail: hendrik.annuth@fh-wedel.de

Betreut von:
Marco Pawłowski
Fachhochschule Wedel
Feldstraße 143
22880 Wedel
Telefon: 04103 - 80 48 - 811
E-Mail: marco.pawlowski@fh-wedel.de

Kurzfassung

Die rasante Entwicklung von *Large Language Models (LLM)*, wie *ChatGPT*, hat dazu geführt, dass aktuelle Modelle Texte erzeugen können, die von menschlich verfassten Texten kaum zu unterscheiden sind. Dies ist mit Risiken verbunden, vor allem in Bezug auf die Verbreitung von Falschinformationen. Um diese Risiken zu minimieren, ist die Entwicklung von Detektoren, welche von *Künstlicher Intelligenz (KI)* generierte Texte identifizieren können, erforderlich. Während moderne Detektoren englischsprachige Texte mit hoher Genauigkeit klassifizieren können, stellt die Erkennung in anderen Sprachen, wie beispielsweise im Deutschen, ein weitgehend unerforschtes Gebiet dar. Ein zusätzliches Problem ist die mangelnde Robustheit aktueller Detektoren. Selbst einfache Manipulationen des zu klassifizierenden Textes können diese Detektoren vor erhebliche Herausforderungen stellen.

Diese Thesis präsentiert einen robusten Detektor zur Erkennung von *KI*-generierten deutschen Texten. Für dessen Training und Evaluation wird der erste deutschsprachige Datensatz im Forschungsbereich erstellt, bestehend aus 70.749 menschlichen und 70.617 *KI*-generierten Texten. Dieser Datensatz, bereichert durch acht verschiedene Textgattungen und sieben unterschiedliche *Prompt*-Vorlagen, ist in seiner Art sprachübergreifend einzigartig. Der Detektor erreicht ein F1-Maß von 97,89% und demonstriert eine hohe Generalisierungsfähigkeit. Weiterhin stellt die Thesis wirksame Maßnahmen vor, die die Erfolgsraten möglicher Angriffe auf ein Minimum reduzieren. Dies legt den Grundstein für zukünftige Forschungen zur Robustheit in der Erkennung von *KI*-generierten Texten.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

API *Application Programming Interface*

ASCII *American Standard Code for Information Interchange*

BERT *Bidirectional Encoder Representation from Transformers*

BPE *Byte-Pair-Encoding*

GLTR *Giant Language Model Test Room Tool*

GPT *Generative Pre-trained Transformer*

KI *Künstliche Intelligenz*

LLaMA *Large Language Model Meta AI*

LLM *Large Language Model*

MLM *Masked Language Modeling*

NLP *Natural Language Processing*

OOD *Out-Of-Distribution*

PaLM *Pathways Language Model*

RLHF *Reinforcement Learning from Human Feedback*

RoBERTa *Robustly Optimized BERT Pretraining Approach*

SOTA *State Of The Art*

TF-IDF *Term Frequency-Inverse Document Frequency*

VPN *Virtual Private Network*

1. Einleitung

1.1. Auswirkungen nicht erkennbarer KI-generierter Texte

> „The rise of powerful AI will be either the best or the worst thing ever to happen to humanity. We do not yet know which.“[1]

Stephen Hawkings Äußerung bei der Eröffnungsfeier des *Cambridge Centre for the Future* beleuchtet die ambivalenten Erwartungen bezüglich der Entwicklung von *KI*. Angesichts jüngster Fortschritte, vor allem durch *LLMs* wie *ChatGPT*, gewinnt *KI* in vielen Alltagsbereichen an Bedeutung. *LLMs* sind in der Lage die Art und Weise, wie wir kommunizieren, illustrieren und kreieren maßgeblich zu verändern. Die einfache Bedienbarkeit und Vielseitigkeit von *LLMs* haben ihren Einsatz in diversen Gesellschaftsbereichen, von Bildung und Medizin bis hin zu Kunst und Unterhaltung, vorangetrieben.

In der Bildung können *LLMs* Pädagogen bei der Erstellung ihres Lehrmaterials helfen, indem sie umfangreiche Literaturbestände analysieren und relevante Inhalte extrahieren.[2] Eine Studie von *Bhat et al.*[3] zeigte, dass Modelle wie *GPT-3* in der Lage sind, Prüfungsfragen zu generieren, die von Experten positiv bewertet werden. Darüber hinaus können *LLMs* den Lernenden maßgeschneiderte Unterstützung bieten und als ständig verfügbare Tutoren dienen.[4] Im medizinischen Bereich können sie zur psychologischen Beratung beitragen[5] und komplexe medizinische Informationen aufbereiten.[6]

---

[1] Hawking: Humanity, 2016.
[2] Kasneci u. a.: Education, 2023, S. 2f.
[3] Bhat u. a., 2022, S. 3.
[4] Dwivedi u. a.: Perspectives, 2023, S. 41.
[5] Dwivedi u. a.: Perspectives, 2023, S. 30.
[6] Jeblick u. a.: Medicine, 2022, S. 13.

Neben ihren Vorteilen bringen *LLMs* auch Risiken und potenziellen Missbrauch mit sich. Ein Hauptproblem ist die Herausforderung, zwischen menschlich verfassten und *KI*-generierten Texten zu unterscheiden.[7] Diese Unklarheit kann es erschweren, authentisches Wissen von nicht überprüften Informationen zu unterscheiden.[8] *NewsGuard* hat beispielsweise 347 *KI*-basierte Nachrichtenportale identifiziert, bei denen die Nachrichtentexte mit minimaler menschlicher Beteiligung erstellt wurden.[9] Zudem besteht die Gefahr, dass Lernende *LLMs* nutzen, um Texte zu generieren und als eigene Arbeit ausgeben.[10] Eine Studie von *Gao et al.*[11] zeigt die Fähigkeit von *LLMs*, überzeugende Forschungszusammenfassungen zu formulieren. Dies stellt eine ernsthafte Bedrohung für die Integrität des Bildungssystems dar und erfordert von Bildungseinrichtungen, sich bewusst mit diesen Technologien auseinanderzusetzen, um die Fairness und Validität von Prüfungen zu wahren.[12] Darüber hinaus könnten *LLMs* zur Verbreitung von Falschinformation oder Hassreden in sozialen Medien beitragen, entweder absichtlich oder aufgrund von einem *Bias* im *LLM*.[13] Schließlich sollte auch das Potenzial von *LLMs* betrachtet werden, bestehende Angriffe zu optimieren. Sie können beispielsweise *Phishing*-Angriffe effektiver machen, indem sie den Schreibstil vertrauenswürdiger Quellen imitieren.[14]

Die rapide Integration von *LLMs* hat zu Reaktionen von verschiedenen Organisationen und Institutionen geführt. So hat das Bildungsministerium von *New York City ChatGPT* aus Schulnetzwerken verbannt.[15] Ebenso hat *Stack Overflow* den Einsatz generativer *KI* bei der Erstellung von Beiträgen verboten, da sie hohe Raten an Fehlinformationen enthalten.[16] Doch die aktuellen Regulierungsansätze sind oft nicht auf *LLMs* zugeschnitten. *Hacker et al.*[17] betonen, dass bestehende Regulierungen von *KI*, insbesondere in der *Europäischen Union*, primär für traditionelle *KI*-Modelle entwickelt wurden.

[7] Susnjak: Exam, 2022, S. 1.
[8] Kasneci u. a.: Education, 2023, S. 7.
[9] Sadeghi u. a.: Misinformation, 2023.
[10] Dwivedi u. a.: Perspectives, 2023, S. 40.
[11] Gao u. a.: Abstracts, 2023, S. 2.
[12] Cotton / Cotton / Shipway: Integrity, 2023, S. 6.
[13] Solaiman u. a.: Impacts, 2019, S. 1; Dwivedi u. a.: Perspectives, 2023, S. 43.
[14] Baki u. a.: Masquerade, 2017, S. 1f.
[15] Elsen-Rooney: NYC, 2023.
[16] o.V.: Banned, 2022.
[17] Hacker / Engel / Mauer: Regulating, 2023, S. 1.

Angesichts dieser Komplexität und der zahlreichen Herausforderungen ist die Erkennung von *KI*-generierten Texten eine Schlüsselvariable im verantwortungsvollen Umgang mit *LLMs*. Dieser Standpunkt wird auch von *OpenAI*, dem Entwickler von *ChatGPT*, unterstrichen.[18]

1.2. Erkennung von KI-generierten Texten

Die rapide Entwicklung der *LLMs* beeinflusst die Erzeugung von *KI*-generierten Texten. Insbesondere *OpenAIs ChatGPT*, welches am 30. November 2022 veröffentlicht wurde,[19] und *Googles Bard*[20] haben sich in diesem Segment als führend etabliert. Dabei ist *ChatGPT* im Juni 2023 mit über 100 Millionen Nutzern[21] das populärste unter den *LLMs* und hat mehr als 30-mal so viele Nutzer wie Bard.[22] Das Modell erreichte die ersten Millionen Nutzer bereits fünf Tage nach seiner Veröffentlichung. Ein Schlüsselfaktor für den Erfolg von *ChatGPT* ist die intuitive Bedienbarkeit. Mittels eines Chats können dem Modell in einer Konversation auf natürliche Weise Anweisungen und Kontextinformationen für verschiedene Aufgaben mitgeteilt werden. Diese Fähigkeit, natürliche Sprache zu verarbeiten, verdankt das Modell vor allem seinem Lernverfahren, dem *Reinforcement Learning from Human Feedback* (*RLHF*). Schon beim Vorgänger *Generative Pre-trained Transformer* (*GPT*)-2 stellten *Solaiman et al.*[23] fest, dass es Menschen schwerfällt, *KI*-generierte Texte von menschlichen Texten zu unterscheiden. Bei der Identifizierung von Texten, die von *ChatGPT* generiert wurden, betrug die Genauigkeit menschlicher Klassifikationen zwischen 50% und 52%. Dabei generiert *ChatGPT* Texte mit einer solchen Originalität, dass selbst Plagiatsdetektoren diese nicht identifizieren können.[24]

Aufgrund dieser Entwicklung ist die Erstellung eines Klassifikationsmodells zur Erkennung von *KI*-generierten Texten von großer Bedeutung. In dieser binären Klassifikationsaufgabe wird ermittelt, ob ein gegebener Text von einem Menschen oder einer *KI* verfasst wurde. Solche Klassifikationsmodelle

[18] Solaiman u. a.: Impacts, 2019, S. 10.
[19] OpenAI: ChatGPT, 2022.
[20] Pichai: Next, 2023.
[21] Duarte: Users, 2023.
[22] Coles: Usage, 2023.
[23] Solaiman u. a.: Impacts, 2019, S. 10.
[24] Khalil / Er: Will, 2023.

werden auch als Detektoren bezeichnet. Man unterscheidet hierbei zwischen *Black-Box*-Detektoren und *White-Box*-Detektoren.[25] *White-Box*-Detektoren nutzen Informationen über die internen Zustände der *LLMs* und benötigen dementsprechend Zugriff auf deren Quellcode. Da aktuelle Modelle, wie *ChatGPT*, in der Regel *Black-Box*-Systeme sind, fokussiert sich diese Arbeit auf die Entwicklung eines *Black-Box*-Detektors, der das Erkennen von *KI*-generierten Texten anhand von Mustern in den Trainingsdaten lernt.

Ein idealer Detektor sollte theoretisch die folgenden fünf Kriterien erfüllen: Genauigkeit, Dateneffizienz, Generalisierbarkeit, Interpretierbarkeit und Robustheit.[26] Der *State Of The Art* (*SOTA*)-Detektor von *Guo et al.*[27] zur Klassifikation von englischsprachigen Texten erzielte beeindruckende Ergebnisse mit einem durchschnittlichen F1-Maß von 98,78% auf einem englischsprachigen Datensatz, wobei deren Datensatz aus 85.449 Texten besteht. Zudem konzipierten sie einen der wenigen Detektoren für nicht-englischsprachige Texte und erzielten bei chinesischen Texten vergleichbare Resultate. Die Forschung bezüglich der Erkennung von nicht-englischsprachigen *KI*-generierten Texten ist noch unterrepräsentiert.

Zusätzlich dazu mangelt es den Detektoren an Robustheit und Generalisierbarkeit. Sowohl einfache Manipulationen, wie das Hinzufügen von Homoglyphen[28] oder das Einfügen von Leerzeichen an bestimmten Stellen im Text[29], als auch komplexe Angriffsmethoden, wie der Einsatz eines Paraphrasierungs-Modells[30], können die meisten Detektoren täuschen. Gegen solche manipulierten Texte wurden bisher keine effektiven Maßnahmen vorgeschlagen. Weiterhin wird die Generalisierbarkeit der Detektoren hinsichtlich bisher ungesehener Textdaten in Frage gestellt. Während *Pegoraro et al.*[31] demonstrierten, dass keiner der analysierten Detektoren in der Lage war, die *KI*-generierten Texte ihres Datensatzes zu identifizieren, bestätigten *Guo et al.*[32] die Wirksamkeit ihres Detektors auf *Out-Of-Distribution* (*OOD*)-Texten.

[25] Cai / Cui: Evade, 2023, S. 1f; Tang / Chuang / Hu: Science, 2023, S. 1.
[26] Jawahar / Abdul-Mageed / Lakshmanan: Automatic, 2020, S. 2296.
[27] Guo u. a.: Close, 2023, S. 12.
[28] Wolff / Wolff: Attacking, 2020, S. 2ff.
[29] Cai / Cui: Evade, 2023, S. 5.
[30] Krishna u. a.: Paraphrasing, 2023, S. 4ff.
[31] Pegoraro u. a.: Chatgpt, 2023, S. 4.
[32] Guo u. a.: Close, 2023, S. 13.

Die Interpretierbarkeit ist ebenfalls ein zentrales Forschungsthema, da die *SOTA*-Detektoren häufig als *Black-Box*-Modelle agieren. In diesem Kontext lieferten *Mitrovic et al.*[33] erste Erkenntnisse mittels *Shapely Additive Explanations*.

1.3. Überblick

Angesichts der Fähigkeit von *LLMs* wie *ChatGPT*, Texte in mehr als 50 verschiedenen Sprachen zu generieren, ist es auffällig, dass Detektionsmodelle vorwiegend auf englischsprachigen Texten trainiert werden. Diese Diskrepanz hat zur Motivation beigetragen, erstmalig die Detektion von Texten in deutscher Sprache zu untersuchen. In dieser Arbeit wird ein Datensatz aus 70.617 *KI*-generierten Texten und 70.749 von Menschen verfassten Texten in deutscher Sprache zusammengestellt. Ziel ist es, einen effizienten Detektor auf Basis dieser Daten zu trainieren und zu evaluieren. Darüber hinaus wird die Generalisierbarkeit des Detektors hinsichtlich unbekannter Textgattungen und *KI*-erzeugten Texten, die mit diversen *Promptstrategien* generiert werden, untersucht. Ein weiterer Schwerpunkt dieser Arbeit ist die Analyse verschiedener Angriffsszenarien. Ausgehend von dieser Analyse werden geeignete Gegenmaßnahmen vorgeschlagen.

Kapitel 2 bietet einen Überblick über den aktuellen Stand der Forschung. Es behandelt sowohl *LLMs*, die für die Generierung von künstlichen Texten benutzt werden, als auch existierende Methoden zu deren Detektion. Im Anschluss daran werden mögliche Manipulationen der Texte diskutiert, die eine Klassifikation als *KI*-generiert umgehen könnten. Abschließend werden Maßnahmen des allgemeinen Bereiches des *Natural Language Processings (NLP)* gegen derartige Manipulationen untersucht, mit dem Ziel, diese Maßnahmen zur robusten Erkennung von *KI*-generierten Texten anzuwenden.

[33] Mitrović / Andreoletti / Ayoub: Chatgpt, 2023.

In Kapitel 3 wird das methodische Vorgehen dargelegt. Besonderes Augenmerk liegt auf der Erstellung eines umfangreichen Datensatzes, der 141.366 Dateneinträge und acht verschiedene Textgattungen umfasst. Die Texte wurden mit Hilfe von *ChatGPT* und unter Verwendung von sieben verschiedenen *Prompt*-Vorlagen generiert. Zudem werden die Implementierungen der Angriffe und Gegenmaßnahmen erörtert.

Kapitel 4 präsentiert die Ergebnisse der sieben durchgeführten Experimente. Diese dienen der Validierung des Detektors und bilden die Grundlage für die Diskussion im darauf folgenden Kapitel.

Die Diskussion in Kapitel 5 evaluiert die Experimente hinsichtlich der Auswahl des Detektormodells, der Generalisierbarkeit auf unbekannte *Prompts* und Textgattungen sowie der Robustheit des Detektors gegenüber Angriffen.

Das abschließende Kapitel fasst die Ergebnisse dieser Thesis zusammen und bietet einen Ausblick auf zukünftige Forschungen im Bereich der Erkennung von *KI*-generierten Texten.

2. Stand der Forschung

In diesem Kapitel werden zunächst die *LLMs*, die zur Textgenerierung genutzt werden, vorgestellt. Anschließend werden bestehende Detektionsmethoden zur Erkennung von *KI*-generierten Texten analysiert. Die Robustheit dieser Detektoren nimmt eine zentrale Rolle ein; daher wird ihre Leistungsfähigkeit im Kontext von Gegenmaßnahmen und potenziellen Angriffen detailliert untersucht. Abschließend wird ein Überblick über die Limitationen und zentralen Erkenntnisse der bisherigen Forschung gegeben, der als Grundlage für die Entwicklung des in dieser Arbeit vorgestellten Detektors dient.

2.1. Large Language Models

LLMs haben in kürzester Zeit zu erheblichen Fortschritten im Bereich des *NLPs* geführt. Sie können für zahlreiche *NLP*-Aufgaben, von der Textklassifizierung bis hin zur Generierung menschenähnlicher Texte, genutzt werden. Im Folgenden wird die Definition sowie der historische Hintergrund der *LLMs* erörtert, um ein tieferes Verständnis ihrer Entwicklung und Relevanz zu gewinnen. Abschließend wird die Architektur der verschiedenen Kategorien von *LLMs* beschrieben.

2.1.1. Definition und Geschichte von LLMs

LLMs sind große Sprachmodelle, die sich durch ihre enormen Größen von mehreren hundert Millionen bis zu mehreren hundert Milliarden Parametern auszeichnen. Sie werden auf umfangreichen, nicht annotierten Textdatensätzen trainiert.[1] Der maßgebliche Erfolg dieser Modelle kann der zugrunde

[1] Zhao u. a.: Language, 2023, S. 2.

liegenden Architektur, dem *Transformer*[2], zugeschrieben werden.[3] Dank seiner Fähigkeit, komplexe Datenmuster zu erkennen und seiner bemerkenswerten Skalierbarkeit, hat sich der *Transformer* als besonders geeignet für die Sprachverarbeitung erwiesen. Die grundlegende Trainingsstrategie der *LLMs* ist das selbstüberwachte Lernen, bei dem das Modell lernt, fehlende Wörter oder Phrasen in einem vorgegebenen Kontext zu ergänzen. Dadurch kann das Modell ein tiefgehendes Verständnis für semantische Beziehungen zwischen Wörtern und dem Kontext in natürlicher Sprache entwickeln.[4] Typischerweise werden zu Beginn Basismodelle mit großen Mengen an Textdaten trainiert, die in der Lage sind, langfristige Abhängigkeiten im Text darzustellen. Für spezifische Anwendungen ist häufig ein *Fine-Tuning* erforderlich, um das Modell optimal auf die jeweilige Aufgabe abzustimmen.[5]

Die Entwicklung aktueller *LLMs* wurde maßgeblich durch die wegweisenden Forschungsarbeiten zweier grundlegender Modelle beeinflusst: Dem *GPT*[6] von *OpenAI* und dem *Bidirectional Encoder Representation from Transformers* (*BERT*)[7] von *Google*. Während *BERT* in einer Vielzahl von *NLP*-Aufgaben, einschließlich der *Named Entity Recognition* und Textklassifizierung, herausragende Leistungen zeigte und die damaligen *SOTA NLP*-Modelle übertraf[8], legte *GPT* den Grundstein für die Entwicklung generativer Textmodelle.

Gegenwärtige *LLMs* können grob in drei architektonische Kategorien unterteilt werden: *Decoder-Only LLMs*, die einer *GPT*-ähnlichen Architektur folgen, *Encoder-Only* Modelle, die der *BERT*-Architektur ähneln, und *Encoder-Decoder LLMs*.[9] Eine historische Analyse, dargestellt in Abbildung 2.1, zeigt, dass *Decoder-Only* Modelle die Entwicklung dominieren. Trotz des ursprünglichen Einflusses von *BERT* scheint das Interesse an *BERT*-ähnlichen Modellen nachgelassen zu haben. Aktuelle Entwicklungen konzentrieren sich vor allem auf die *Decoder-Only* Architektur. Parallel dazu setzt *Google* die Forschung zu *Encoder-Decoder LLMs* aktiv fort. Dieser Trend hat dazu beigetragen, dass *OpenAI* eine dominante Position in der *LLM*-Forschung einnimmt und

[2] Vaswani u. a.: Attention, 2017, S. 10.
[3] Min: Models, 2021, S. 1ff.
[4] Min: Models, 2021, S. 2.
[5] Radford u. a.: Generative, 2018, S. 1f; Zhao u. a.: Language, 2023, S. 36.
[6] Radford u. a.: Generative, 2018.
[7] Devlin u. a.: Bert, 2018.
[8] Devlin u. a.: Bert, 2018, S. 6f.
[9] Yang u. a.: Survey, 2023, S. 2.

viele Konkurrenten Schwierigkeiten haben, ähnlich leistungsfähige Modelle zu entwickeln.[10] Ein weiteres bemerkenswertes Detail in Abbildung 2.1 ist, dass hauptsächlich große Technologieunternehmen die Entwicklung von *LLMs* vorantreiben. Dies lässt sich auf die enormen Rechenanforderungen zurückführen, die mit der Entwicklung dieser Modelle verbunden sind.[11]

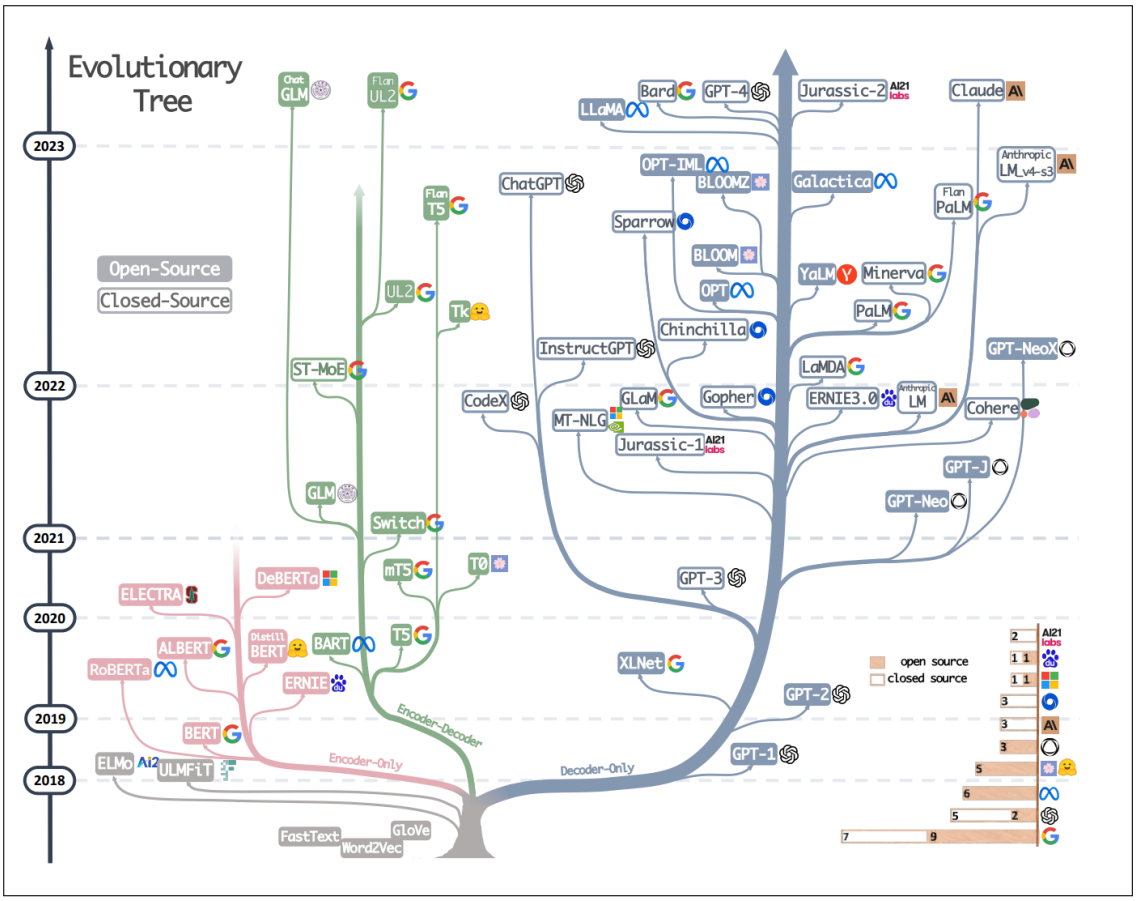

Abbildung 2.1.: Evolution der *LLMs*.[12]

Bei der Skalierung kleinerer Modelle wie *GPT-1* stießen Entwickler auf das Phänomen der sogenannten emergenten Fähigkeiten. *Wei et al.*[13] definieren diese als Fähigkeiten, die bei kleineren Modellen nicht manifestiert sind und erst ab einer bestimmten Modellgröße erlernt werden. Interessanterweise lassen sich diese emergenten Fähigkeiten nicht direkt durch das Skalierungsgesetz antizipieren, sondern scheinen erst nach Erreichen eines spezifischen Schwellenwerts signifikant über den Erwartungswert hinauszugehen. Ein

[10] Yang u. a.: Survey, 2023, S. 3.
[11] Yang u. a.: Survey, 2023, S. 3; Rudolph / Tan / Tan: Chatbots, 2023, S. 368f.
[12] Yang u. a.: Survey, 2023, S. 3
[13] Wei u. a.: Emergent, 2022, S. 2f.

Beispiel stammt von *Chowdhery et al.*[14], die während der Entwicklung des *Pathways Language Model* (*PaLM*) feststellten, dass das Modell erst nach einer bestimmten Skalierungsgrenze in der Lage war, kontextbezogene Sprichwörter adäquat vorherzusagen. Eine weiterführende Erkenntnis in diesem Bereich lieferten *Brown et al.*[15], die beobachteten, dass *GPT-3* Fähigkeiten für kontextuelles Lernen aufweist. Insbesondere konnte das Modell sogenannte *Few-Shot-Aufgaben* bewältigen, indem es nur wenige Beispiele für eine bestimmte Aufgabe benötigte. Dieser Ansatz, der als *prompt-basiertes* Lernen[16] bekannt ist, ermöglicht es, vortrainierte Sprachmodelle durch gezielte Eingabeaufforderungen auf spezifische Aufgabenstellungen auszurichten.

Ein bedeutender Fortschritt in der Entwicklung von generativen Sprachmodellen wurde durch das Modell *InstructGPT* von *OpenAI* gemacht. *Ouyang et al.*[17] präsentierten einen Ansatz, der es *LLMs* ermöglicht, menschliche Anweisungen präziser zu befolgen. Dieses Verfahren beruht auf zwei zentralen Mechanismen: *Instruction Tuning* und *RLHF*. Während das *Instruction Tuning*[18] ein *Fine-Tuning* des Modells anhand eines selektierten Datensatzes von Anweisungen und zugehörigen korrekten Antworten vorsieht, fokussiert *RLHF*[19] sich auf das Training eines Belohnungsmodells, welches menschliche Präferenzen widerspiegelt. Dieses Belohnungsmodell dient dazu, *LLMs* in iterativen Schritten zu optimieren.

Die Entwicklung der *LLMs* hat in den letzten Jahren eine wachsende Tendenz zur Nicht-Offenlegung von Quellcode und der Modellarchitekturen gezeigt. Obwohl *OpenAI* ursprünglich als gemeinnützige Organisation gegründet wurde, hat es seinen Fokus in Richtung kommerzieller Interessen verlagert, was Bedenken hinsichtlich seiner ursprünglichen Verpflichtung zur Offenheit aufwirft[20]. Noch bis zum Jahr 2020 waren die meisten dieser Modelle als *Open-Source* veröffentlicht und der wissenschaftlichen Gemeinschaft frei zugänglich.

[14] Chowdhery u. a.: PaLM, 2022, S. 16.
[15] Brown u. a.: Few-shot, 2020, S. 3f.
[16] Min: Models, 2021, S. 10.
[17] Ouyang u. a.: Feedback, 2022.
[18] Ouyang u. a.: Feedback, 2022, S. 3, 9.
[19] Ouyang u. a.: Feedback, 2022, S. 2ff.
[20] Rudolph / Tan / Tan: ChatGPT, 2023, S. 343f.

Trotz der Tendenz zur Nicht-Veröffentlichung bestimmter Modelle existieren weiterhin bemerkenswerte *Open-Source*-Alternativen zu den vorherrschenden *LLMs* von *Google* und *OpenAI*. Ein prominentes Beispiel hierfür ist *Large Language Model Meta AI* (*LLaMA*), eine Sammlung von Sprachmodellen, die zwischen sieben Milliarden und 65 Milliarden Parametern variieren. Seit dem 18. Juli 2023 steht *LLaMA 2* der Öffentlichkeit zur Verfügung. In diversen *Benchmark*-Analysen hat *LLaMA 2* alle existierenden *Open-Source*-Modelle übertroffen und kann sogar mit einigen der *Closed-Source*-Modelle mithalten[21]. Dennoch gibt es eine deutliche Leistungsdifferenz zwischen *LLaMA 2* und den *SOTA*-Modellen wie *GPT-4* und *PaLM-2-L*[22].

2.1.2. Architektur

Nachdem bereits erörtert wurde, dass die *Transformer*-Architektur eine zentrale Rolle in der Entwicklung von *LLMs* spielt, wird in diesem Kapitel ihre detaillierte Struktur und Funktionsweise in den Sprachmodellen betrachtet. *Vaswani et al.*[23] initiierten mit dem *Transformer* einen Paradigmenwechsel in der sequenziellen Datenverarbeitung, insbesondere in der Textverarbeitung. Vor dem *Transformer* wurden hauptsächlich *rekurrente neuronale Netzwerke* und *Long Short-Term Memory Netzwerke* eingesetzt. Der *Transformer* brachte eine verbesserte Parallelisierung und dadurch verkürzte Trainingszeiten.

Im Wesentlichen besteht der *Transformer* aus einem *Encoder*, der Eingabedaten in für das Modell interpretierbare Vektoren umwandelt, und einem *Decoder*, der diese Vektoren in die gewünschte Ausgabe übersetzt. Eine Schlüsselkomponente dieser Architektur ist der *Self-Attention*-Mechanismus, der es ermöglicht, Abhängigkeiten in Daten unabhängig von ihrem Abstand in der Sequenz zu modellieren. Dies unterscheidet den *Transformer* von konventionellen, sequenziellen Datenverarbeitungsmodellen, die Schwierigkeiten haben, weit entfernte Abhängigkeiten zu modellieren. Der *Self-Attention*-Mechanismus gewichtet jedes Eingabeelement nach seiner Relevanz für die folgende Ausgabe.

[21] Touvron u. a.: LLaMA, 2023, S. 31.
[22] Touvron u. a.: LLaMA, 2023, S. 8.
[23] Vaswani u. a.: Attention, 2017, S. 1ff.

Der *Multi-Head-Self-Attention*-Prozess, eine Weiterentwicklung in *LLMs*, führt diese Gewichtung gleichzeitig in mehreren parallelen *Köpfen* durch, wodurch umfassendere Erkenntnisse aus den Daten gewonnen werden können.

Ein weiterer zentraler Aspekt der Funktionsweise dieser Modelle ist das Verfahren zur Textverarbeitung. Statt Sätze wortweise zu analysieren, segmentieren die Modelle den Text in individuelle *Tokens*. Je nach angewandter *Tokenisierungsmethode* können diese *Tokens* vollständige Wörter, Subwörter oder sogar einzelne Zeichen repräsentieren.[24] Moderne Modelle wie *LLaMA 2* und *GPT-4* verwenden das *Byte-Pair-Encoding* (*BPE*) zur *Tokenisierung*, welches iterativ die am häufigsten benachbarten Zeichen zu *Tokens* zusammenfasst.[25]

In Abbildung 2.2 wird der *Tokenisierungsprozess* von *GPT-3* veranschaulicht. Zeichen, die benachbart sind und die gleiche Farbe haben, werden zu einem einzelnen *Token* zusammengefasst. Jeder dieser *Tokens* wird intern durch einen eindeutigen Identifikator repräsentiert, der auf den entsprechenden *Token* im Wortschatz von *GPT-3* verweist. Das Vokabular von *GPT-3* besteht aus insgesamt 50.257 einzigartigen *Tokens*.[26]

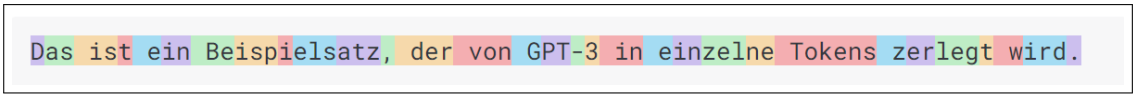

Abbildung 2.2.: Tokenisierung eines Beispielsatzes durch *GPT-3*. [27]

Yang et al.[28] und *Min et al.*[29] beschreiben die drei *LLM*-Kategorien wie folgt:

Decoder-Only: Modelle der *Decoder-Only* Kategorie zeichnen sich durch ihre *autoregressive* Struktur aus. Konkret bedeutet dies, dass die Vorhersage des folgenden *Tokens* einer eingegebenen Sequenz auf Basis der vorherigen *Tokens* erfolgt. Das Trainingsverfahren ist darauf ausgelegt, den *Token* mit der höchsten Wahrscheinlichkeit auszugeben.

[24] Zhao u. a.: Language, 2023, S. 15f.
[25] Zhao u. a.: Language, 2023, S. 15.
[26] Radford u. a.: Generative, 2018, S. 2ff.
[27] OpenAI: Tokenizer, 2023
[28] Yang u. a.: Survey, 2023, S. 2ff.
[29] Min: Models, 2021, S. 4f.

Encoder-Only: Diese Modelle arbeiten *bidirektional*, was ihnen ermöglicht, sowohl vorhergehende als auch die nachfolgende *Tokens* in ihren Vorhersagen zu berücksichtigen. Die Anwendung dieses Ansatzes zeigt sich vor allem im Trainingsverfahren, bei welchem die Methode des *Masked Language Modeling* (*MLM*) verwendet wird. In diesem Prozess werden *Tokens* zufällig maskiert und das Modell muss basierend auf dem umliegenden Kontext die maskierten *Tokens* korrekt vorhersagen.

Encoder-Decoder: Die *Encoder-Decoder* Modelle zeichnen sich durch ihre Fähigkeit aus, nicht nur einzelne *Tokens*, sondern vollständige *Token*-Sequenzen in einem Durchlauf zu generieren. Während des Trainings werden modifizierte Sequenzen zugeführt. Die Aufgabe des Modells besteht dann darin, die ursprüngliche Sequenz durch Umkehrung der vorgenommenen Modifikationen zu rekonstruieren. Diese Modifikationen können beispielsweise die Permutation von Sätzen oder das Entfernen von *Tokens* umfassen. Aufgrund ihrer speziellen Architektur sind diese Modelle besonders geeignet für Aufgaben wie maschinelles Übersetzen oder automatisierte Textzusammenfassungen.

Die spezifischen Trainingsmethoden und Funktionsweisen der drei Architekturkategorien sind in Abbildung 2.3 dargestellt.

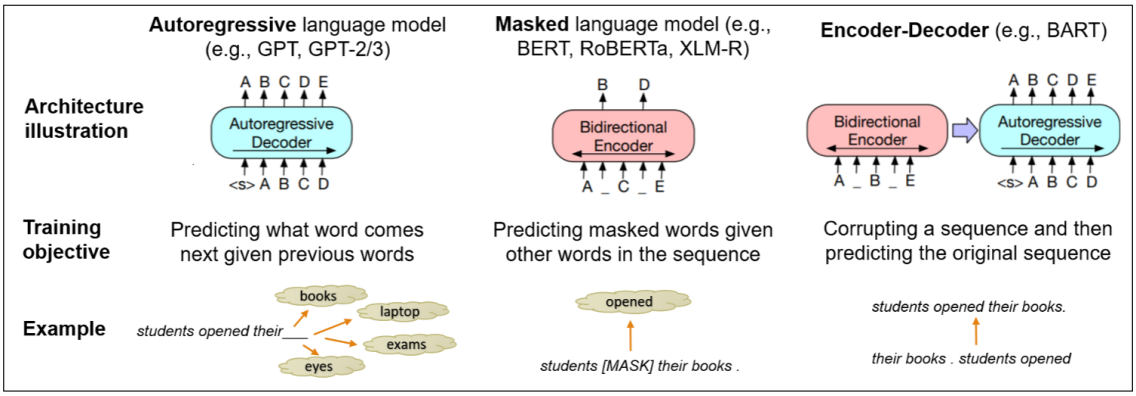

Abbildung 2.3.: Trainingsmethoden der verschiedenen *LLM*-Architekturen. [30]

[30] Min: Models, 2021, S. 4

2.2. Erkennung von KI-generierten Texten

In diesem Kapitel wird die Problemstellung zur Identifizierung von *KI*-generierten Texten dargelegt. Zunächst wird eine präzise Definition der Anforderungen vorgestellt, die an entsprechende Detektionsmechanismen gestellt werden. Anschließend erfolgt eine kategorische Übersicht über die aktuellen Methoden zur Erkennung von *KI*-generierten Texten.

2.2.1. Anforderungen an den Detektor

Ein idealer Detektor sollte in der Theorie die folgenden fünf Kriterien[31] erfüllen:

Genauigkeit: Das Ziel ist eine minimale Rate an Falsch-Positiven und Falsch-Negativen. Besonders Falsch-Positive, bei denen menschlich erstellte Texte als *KI*-generiert eingestuft werden, könnten je nach Anwendungsgebiet, selbst bei geringer Rate, problematisch sein, da potenziell bestimmte Autorenstile fehlinterpretiert und somit Meinungen bestimmter Gruppen unterdrückt werden könnten.[32]

Dateneffizienz: Die Klassifizierung sollte sich mit einer minimalen Datenmenge als zuverlässig erweisen.[33]

Generalisierbarkeit: Es ist essentiell, dass der Detektor mit diversen Modellierungen von *LLMs* zurechtkommt, einschließlich unterschiedlicher Trainingsdaten, Architekturen und Parameter. Eine schnelle Adaption an neue *LLMs* ist notwendig.[34]. Dies lässt die Perspektive eines fortwährenden Wettrennens, einem „Cat and mouse game“[35], erahnen.

Interpretierbarkeit: Die Klassifizierungsentscheidungen sollten transparent und für den Nutzer nachvollziehbar sein.[36]

[31] Jawahar / Abdul-Mageed / Lakshmanan: Automatic, 2020, S. 2296.
[32] Solaiman u. a.: Impacts, 2019, S. 16.
[33] Zellers u. a.: Defending, 2019, S. 3.
[34] Uchendu u. a.: Authorship, 2020, S. 3; Solaiman u. a.: Impacts, 2019, S. 10.
[35] Solaiman u. a.: Impacts, 2019, S. 10.
[36] Gehrmann / Strobelt / Rush: Gltr, 2019, S. 2.

Robustheit: Der Detektor sollte auch bei *Adversarial Examples* eine verlässliche Performance aufweisen.[37] *Adversarial Examples* sind gezielt modifizierte Datenpunkte, die darauf abzielen, das Modell zu falschen Vorhersagen zu verleiten. Idealerweise sind die Modifikationen so geschickt, dass diese von Menschen nicht explizit wahrgenommen werden.

Zusätzlich zu den obigen Kriterien fügen *Fröhling et al.*[38] die Anforderung an Zugänglichkeit hinzu. Modelle zur Erkennung von *KI*-generierten Texten sollten, in Anbetracht der enormen Relevanz von *LLMs*, breit zugänglich und leicht konfigurierbar sein.

2.2.2. Methoden

Dieses Kapitel bietet einen umfassenden Überblick über die aktuellen Methoden bezüglich der Differenzierung zwischen Texten, die von *KI* generiert wurden, und solchen, die von Menschen verfasst wurden. Um einen strukturierten Einblick in die bestehende Literatur zu gewährleisten, werden die Detektionsansätze basierend auf den jeweils angewandten Methoden kategorisiert.

Es ist relevant zu berücksichtigen, dass moderne *LLMs*, wie etwa *ChatGPT*, zumeist als *Closed-Source*-Systeme operieren. Vor diesem Hintergrund liegt der Fokus dieser Arbeit insbesondere auf *Black-Box*-Detektoren. Dabei orientiert sich die vorliegende Arbeit an der Klassifikation von *OpenAI*[39], welche die unterschiedlichen Methoden in drei zentrale Kategorien unterteilt. Darüber hinaus wird in dieser Arbeit auch die merkmalsbasierte Erkennung, gemäß *Fröhling et al.*[40], in Betracht gezogen und analysiert.

Einfache Detektoren

Innerhalb der ersten Kategorie befinden sich einfache Detektoren, die ohne spezifisches Vorwissen oder vorausgehendes Training direkt auf die Klassifikation der Textdaten trainiert werden. Diese Klassifikatoren weisen eine

[37] Krishna u. a.: Paraphrasing, 2023, S. 1.
[38] Fröhling / Zubiaga: Feature, 2021, S. 3.
[39] Solaiman u. a.: Impacts, 2019, S. 12.
[40] Fröhling / Zubiaga: Feature, 2021.

geringe Parameterkomplexität und unkomplizierte Handhabung auf. Ein Beispiel hierfür ist der von *Solaiman et al.*[41] vorgeschlagene logistische Regressionsklassifikator, welcher den Text durch *Term Frequency-Inverse Document Frequency* (*TF-IDF*)-Gewichtungen interpretiert. Abhängig von den Einstellungen des verwendeten *LLMs*, erreichten sie bei einem Datensatz, bestehend aus *GPT-2* Ausgaben und WebText Beispielen[42], eine Genauigkeit zwischen 74% und 97%. *Islam et al.*[43] stellten nur geringe Unterschiede zwischen den genutzten Klassifikationsmodellen des maschinellen Lernens fest, wobei der weit verbreitete Ansatz des *Random Forest* eines der besten Ergebnisse erzielte.

Während einfache Detektionsmethoden in Bezug auf Zugänglichkeit und Implementierung punkten, sind sie in ihrer Anpassungsfähigkeit oft eingeschränkt. Die eingeschränkte Generalisierbarbeit dieser Methode erfordert eine spezifische Ausrichtung auf neue Anwendungsbereiche. Dieser Anpassungsprozess kann erheblichen zeitlichen und finanziellen Aufwand mit sich bringen.[44]

Zero-Shot Detektoren

Jawhar et al.[45] definieren die *Zero-Shot* Detektoren als ein vortrainiertes *LLM*, das den Generierungsprozess repliziert, um so die Gesamtwahrscheinlichkeit der zu untersuchenden Textsequenz zu berechnen. Eine Textsequenz wird als *KI*-generiert eingestuft, sofern ihre berechnete Wahrscheinlichkeit stärker dem Durchschnitt der Wahrscheinlichkeiten für *KI*-generierte Texte entspricht als jenem Durchschnitt, der für menschlich verfasste Texte ermittelt wurde. Dieser Ansatz setzt auf die Erkenntnis, dass *LLMs* bei der Generierung des nachfolgenden *Tokens* aus häufig auftretenden *Token*-Bereichen wählen.

Das *Giant Language Model Test Room Tool* (*GLTR*)[46] ergänzt diesen Ansatz dadurch, dass dieses Modell zusätzlich zur Wahrscheinlichkeit jedes *Tokens* den absoluten Rang der Tokens für die Detektion nutzt. Jedes Token wird dann

[41] Solaiman u. a.: Impacts, 2019, S. 12f.
[42] OpenAI: Output, 2019.
[43] Islam u. a.: Distinguishing, 2023, S. 5.
[44] Fröhling / Zubiaga: Feature, 2021, S. 5.
[45] Jawahar / Abdul-Mageed / Lakshmanan: Automatic, 2020, S. 2301.
[46] Gehrmann / Strobelt / Rush: Gltr, 2019, S. 1.

in eine der folgenden Klassen eingeteilt: *Top-10*, *Top-100*, *Top-1000* und *Platz 1000+*. Eine visuelle Repräsentation dieser Einteilung kann in Abbildung 2.4 eingesehen werden. Die Abbildung 2.4 zeigt deutlich, dass *ChatGPT* dazu neigt, *Tokens* aus den wahrscheinlichsten Bereichen der Verteilung zu verwenden, wie durch die Dominanz der grün markierten Tokens ersichtlich wird.

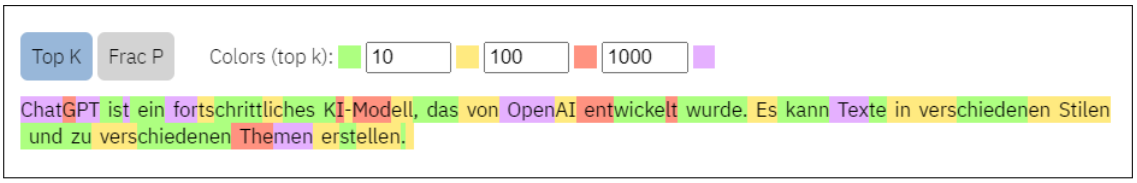

Abbildung 2.4.: Visualisierung der *GLTR*-Klassen eines Satzes.
Die Wahrscheinlichkeiten der *Tokens* werden durch *GPT-2* berechnet. Grün hervorgehobene Segmente kennzeichnen die *Tokens*, die zu den zehn am häufigsten ausgewählten gehören. Tokens, die innerhalb der *Top-100* Rangliste liegen, sind gelb markiert, während jene aus den *Top-1000* in rot dargestellt werden. Lila gekennzeichnete *Tokens* repräsentieren alle weiteren. [47]

Ippolito et al.[48] untersuchten den Einfluss der Klassenverteilung auf die Genauigkeit und stellten eine Verbesserung fest, als sie anstelle von vier Klassen, 50 Klassen gleichmäßig über die potenziellen Rangpositionen eines Vokabulars von 50.000 Wörtern verteilten.

Insgesamt sind *Zero-Shot*-Detektoren eine vielversprechende Methode, jedoch setzt eine optimale Performance dieses Ansatzes die Zugänglichkeit zu dem generierenden Modell oder eines gleichwertig leistungsfähigen Modells voraus.

Zero-Shot Detektoren beruhen auf der Annahme, dass die zu untersuchende Sequenz von einem ähnlichen Modell stammt. Die Herausforderung dabei ist der Zugriff auf solch ein *LLM* und die erforderliche Rechenleistung.

Fine-Tuning von Large Language Models

Die Feinabstimmung von bereits vortrainierten *LLMs*, hat sich als effektiver Ansatz zur Erkennung von *KI*-generierten Texten herausgestellt.

[47] OpenAI: Language, 2019
[48] Ippolito u. a.: Automatic, 2020, S. 1811.

Solaiman et al.[49] führten in ihrer Studie einen solchen Feinabstimmungsprozess am *Robustly Optimized BERT Pretraining Approach* (*RoBERTa*)-Modell durch, welches auf der Architektur von *BERT* basiert. Dieser Detektor erzielte eine beeindruckende Genauigkeit von 95% bei der Unterscheidung zwischen Texten, die von *GPT-2* generiert wurden, und menschlich erzeugten Texten. Zudem stellten sie fest, dass *Encoder-Only LLMs* für die Klassifikation von Texten besser geeignet sind als die anderen beiden Kategorien der *LLMs*. Weitere Studien, wie die von *Fagni et al.*[50] und *Uchendu et al.*[51], bekräftigen die Überlegenheit des *RoBERTa*-Modells gegenüber anderen *LLMs* wie *BERT*, *XLNet* und *DistilBERT* in Bezug auf diese spezifische Detektionsaufgabe.

Für diesen Ansatz werden dem bereits trainierten Basismodell, wie zum Beispiel *RoBERTa*, zusätzliche Schichten für die Klassifikation hinzugefügt. *LLMs* geben eine Sequenz an Vektoren aus, wobei jeder Vektor die Ausgabe für ein bestimmtes *Token* in der Eingabesequenz darstellt. Diese Sequenz wird mittels *Pooling* auf einen einzigen Vektor reduziert, um diesen als Eingabe für die nachgelagerte Klassifikation zu nutzen.[52] Modelle wie *BERT* und *RoBERTa* fügen am Anfang einer Eingabesequenz das sogenannte *[CLS]-Token* ein. *Devlin et al.*[53] bezeichnen dieses Token als *spezielles Klassifikationstoken*, da dieses *Token* durch den Trainingsprozess eine Repräsentation der gesamten Sequenz darstellt. Deshalb hat sich die Extraktion des *[CLS]-Tokens* als gängige Methode beim *Pooling* für Klassifikationsaufgaben etabliert.[54] Nachdem die *Pooling*-Schicht den Ausgang auf eine feste Größe reduziert hat, übernimmt ein vollständig verbundenes neuronales Netzwerk die endgültige Klassifikation. In der Regel wird *Softmax* als Aktivierungsfuktion genutzt, um die Wahrscheinlichkeiten der Klassen vorherzusagen.

Die Wirksamkeit dieser Methode beschränkt sich nicht nur auf einzelne Anwendungsgebiete oder Textgenres. *Mitrovic et al.*[55] nutzten die Feinabstimmungsmethodik ebenfalls erfolgreich, um *KI*-generierte Kurztexte wie Online-Rezensionen mit einer beeindruckenden Genauigkeit von 98% zu erkennen.

[49] Solaiman u. a.: Impacts, 2019, S. 13ff.
[50] Fagni u. a.: Tweepfake, 2021, S. 10.
[51] Uchendu u. a.: Authorship, 2020, S. 8390.
[52] Fagni u. a.: Tweepfake, 2021, S. 8f.
[53] Devlin u. a.: Bert, 2018, S. 4.
[54] Fagni u. a.: Tweepfake, 2021, S. 8f.
[55] Mitrović / Andreoletti / Ayoub: Chatgpt, 2023, S. 6.

Darüber hinaus haben *Fagni et al.*[56] in der Erkennung von *KI*-generierten Tweets die Effektivität dieses Ansatzes verifiziert.

Es ist bemerkenswert, dass auch Texte der *SOTA*-Modelle wie *ChatGPT* durch feinabgestimmte *LLMs* erfolgreich erkannt werden können. In ihrer Untersuchung erreichten Guo et al.[57] mit der Feinabstimmungsmethodik ein F1-Maß von über 98% bei der Unterscheidung von *ChatGPT-generierten* Texten gegenüber menschlich erstellten Texten. Dies wurde auf einem umfangreichen Datensatz, dem *Human ChatGPT Comparison Corpus*, durchgeführt, der sowohl menschliche als auch *KI*-generierte Antworten auf insgesamt 24.000 Fragen enthält. Dies unterstreicht die vielseitige Anwendbarkeit und Übertragbarkeit des Feinabstimmungsansatzes, unabhängig von der Komplexität und dem Fortschrittsgrad des zu untersuchenden Modells. In diesem Rahmen verbesserten *Guo et al.*[58] die Performance des Detektors dadurch, dass sie die Trainingsdaten in einzelne Sätze segmentieren und der Detektor diese einzeln klassifiziert.

Trotz der beeindruckenden Erkennungsgenauigkeiten der Modelle gibt es bedeutende Herausforderungen und Bedenken. Da diese Modelle *Black-Box*-Systeme sind, mangelt es diesen Detektoren an Transparenz und Nachvollziehbarkeit. Jedoch konnten *Mitrovic et al.*[59] aus den Ergebnissen dieser Detektoren bereits erste interpretierbare Informationen ableiten.

Merkmalsbasierte Erkennung

Die merkmalsbasierte Erkennung zur Unterscheidung von Texten, die entweder von Menschen oder Maschinen erzeugt wurden, baut auf der Grundannahme auf, dass zwischen diesen beiden Textklassen signifikante Unterschiede existieren.[60] Die Differenzierungsmerkmale zwischen den beiden Klassen werden durch den Einsatz statistischer Methoden identifiziert[61]. In diesem Identifikationsprozess werden mithilfe von Techniken des *NLPs* aussagekräftige Merkmalsvektoren aus den Textdaten extrahiert.

[56] Fagni u. a.: Tweepfake, 2021, S. 8.
[57] Guo u. a.: Close, 2023, S. 3, S. 12.
[58] Guo u. a.: Close, 2023, S. 14.
[59] Mitrović / Andreoletti / Ayoub: Chatgpt, 2023, S. 3f.
[60] Ma u. a.: AI, S. 14.
[61] Fröhling / Zubiaga: Feature, 2021, S. 8.

Nachdem diese Merkmalsvektoren erstellt wurden, werden sie mit bekannten Klassifikationsalgorithmen weiterverarbeitet. Zu den in diesem Kontext verwendeten Methoden gehören *Support Vector Machines*, *Random Forest* Algorithmen und neuronale Netzwerke. Jede dieser Techniken besitzt das Potenzial, die extrahierten Merkmale in präzise Klassifikationsergebnisse zu überführen.

Fröhling et al.[62] formulieren vier charakteristische Kriterien, in denen sich *KI*-generierte Texte von menschlich verfassten Texten unterscheiden:

Begrenzte Variation in der Wortwahl: Während Menschen in ihrer schriftlichen Kommunikation vielfältige Ausdrucksweisen verwenden, neigen *KI*-Modelle dazu, weniger Synonyme oder Referenzen für Entitäten zu generieren.[63] Dieser Unterschied kann durch die Betrachtung der Anzahl *benannter Entitäten* im Text bemessen werden. Zudem weisen *Guo et al.*[64] darauf hin, dass sich die Verteilung der Wortarten zwischen menschlichen und *KI*-generierten Texten unterscheiden.

Wiederholungen: *Ippolito et al.*[65] zeigen, dass maschinell generierte Texte eine Tendenz zur übermäßigen Nutzung häufiger Wörter und wiederkehrender Phrasen aufweisen. Dies lässt sich durch statistische Analysen wie den Anteil von Stopp-Wörtern oder die Anzahl einzigartiger Wörter erfassen. Nach den Ergebnissen von *Galle et al.*[66] treten *n-Gramme* höherer Ordnung in maschinell generierten Texten häufiger auf.

Kohärenzmängel: Obwohl maschinell generierte Texte häufig flüssig erscheinen, können sie inhaltliche Mängel oder fehlende Zusammenhänge aufweisen.[67] *Fröhling et al.*[68] quantifizieren dies mittels einer *Entity-Grid*-Darstellung, die das Auftreten und die grammatikalische Funktion von Entitäten über verschiedene Sätze verfolgt.

[62] Fröhling / Zubiaga: Feature, 2021, S. 7ff.
[63] Gehrmann / Strobelt / Rush: Gltr, 2019, S. 5.
[64] Guo u. a.: Close, 2023, S. 8.
[65] Ippolito u. a.: Automatic, 2020, S. 1814.
[66] Gallé u. a.: Unsupervised, 2021, S. 1.
[67] Brown u. a.: Few-shot, 2020, S. 33.
[68] Fröhling / Zubiaga: Feature, 2021, S. 8.

Ziellosigkeit: Maschinell generierten Texten mangelt es häufig an einem klaren Ziel oder einer Intention, da Sprachmodelle nicht von menschlichen Bedürfnissen oder Emotionen geleitet werden.[69]

2.3. Robustheit von Detektoren gegenüber manipulierten Texten

Eine der größten Herausforderungen beim Erkennen von *KI*-generierten Texten ist die Robustheit des Detektors. Wie in Kapitel 2.2.2 dargelegt, weist jede der vorgestellten Methoden potenzielle Schwachstellen gegenüber bestimmten Angriffen auf, wodurch ihre Performance in realen Anwendungsumgebungen beeinträchtigt werden kann.[70] Insbesondere in Szenarien, in denen das *LLM* zum Generieren schädlicher Texte eingesetzt wird, besteht ein hohes Risiko, dass Angreifer den Detektor mittels gezielter Attacken umgehen. Deshalb ist es von zentraler Bedeutung, sich intensiv mit möglichen Angriffsszenarien auseinanderzusetzen.

Entwickler müssen sich auch darüber im Klaren sein, dass die Veröffentlichung eines Detektors Angreifern potenziell Hilfestellung bietet, diesen zu umgehen.[71]

Eine Attacke wird als erfolgreich betrachtet, wenn sie nicht nur die Performance des Detektors beeinträchtigt, sondern auch die Textqualität des generierten Textes beibehält.[72] So könnte eine verringerte Textqualität beispielsweise Angriffe weniger effektiv machen. Ein gutes Beispiel hierfür ist eine *Phishing-E-Mail*, die vorgibt, von einer Bank zu stammen; für einen Empfänger wirkt sie weniger überzeugend, wenn sie durch eine ungewöhnliche Wortwahl und viele Tippfehler ins Auge sticht.[73]

[69] Fröhling / Zubiaga: Feature, 2021, S. 9.
[70] Pu u. a.: Deepfake, 2022, S. 13.
[71] Solaiman u. a.: Impacts, 2019, S. 13.
[72] Pu u. a.: Deepfake, 2022, S. 8.
[73] Crothers / Japkowicz / Viktor: Machine-generated, 2023, S. 27.

2.3.1. Angriffsmethoden

Die betrachteten Angriffsmethoden versuchen, durch syntaktische oder semantische Manipulation der *KI*-generierten Texte die Erkennung eines Detektors zu umgehen. Die Literatur identifiziert folgende Angriffsstrategien:

Austausch von Schlüsselwörtern durch Synonyme: *Pu et al.*[74] stellten fest, dass insbesondere Detektoren, die auf der *Transformer*-Architektur basieren, bestimmte Wörter besitzen, die maßgeblich zur Klassifizierung beitragen. Dies ist darauf zurückzuführen, dass *LLMs* den nächsten zu generierenden *Token* anhand seiner Wahrscheinlichkeit auswählen. Mithilfe von *GLTR* können Wörter identifiziert werden, die *Tokens* enthalten, welche an genau dieser Position mit einer hohen Wahrscheinlichkeit auftreten. Diese Wörter werden dann durch Synonyme ersetzt, ohne die Bedeutung des Satzes zu verändern. Die von ihnen vorgestellte Angriffsmethode erwies sich auch ohne zusätzliche Informationen über die Erkennungsverfahren als erfolgreich und konnte in 23,2% bis 91,3% der Versuche die Erkennung umgehen. Es ist jedoch anzumerken, dass die Effizienz des Angriffs wahrscheinlich abnimmt, wenn der Angreifer keinen direkten Zugriff auf das generierende Sprachmodell besitzt, da er in einem solchen Szenario die genauen Wahrscheinlichkeiten der *Tokens* nicht ermitteln kann. Bei diesem Angriff muss ein Gleichgewicht bei der Auswahl der zu modifizierenden Wörter gefunden werden. Zu viele Änderungen könnten die Qualität des Textes beeinträchtigen, während zu wenige möglicherweise nicht ausreichend sind, um den Detektor erfolgreich zu täuschen.

Einfügen von Rechtschreibfehlern: *Gao et al.*[75] zeigten durch ihren *Deep-WordBug*-Ansatz, dass das Einfügen von Rechtschreibfehlern in *KI*-generierten Texten die Performance von *Deep-Learning*-Klassifikationsmodelle signifikant beeinträchtigen können. Fehlerhaft geschriebene Wörter werden von diesen Modellen oft als unbekannt kategorisiert, da sie diese *Tokens* nicht in ihrem Vokabular wiederfinden. Im Gegensatz dazu sind Menschen in der Lage, das ursprüngliche Wort zu identifizieren.

[74] Pu u. a.: Deepfake, 2022, S. 8f.
[75] Gao u. a.: Evade, 2018, S. 10.

Paraphrasierung: Die Forschungsarbeit von *Krishna et al.*[76] hat gezeigt, dass durch das Paraphrasieren von *KI*-generierten Texten die Erkennungsgenauigkeit von Detektoren gesenkt werden kann, ohne die Bedeutung des ursprünglichen Satzes zu beeinträchtigen. In ihrem Experiment verwendeten sie einen *Encoder-Decoder Transformer*, konkret das vortrainierte *T-5 Modell*. Durch *Fine-Tuning* wurde dieser Paraphrasierer namens *Dipper* optimiert. Überraschenderweise gelang es *Dipper*, fortschrittliche Detektoren wie *DetectGPT*[77], *GPTZero*[78] und *OpenAIs Detektor*[79] zu täuschen. Die Effizienz des Paraphrasierens besteht darin, dass es die statistischen Eigenschaften von *KI*-generierten Texten verändern kann. Dennoch stellten *Mitrovic et al.*[80] eine geringfügige Abnahme in der Genauigkeit durch das Paraphrasieren fest, die Genauigkeit lag bei 79%.

Prompt Engineering: Ein *Prompt* ist ein Eingabetext, der an ein Sprachmodell gegeben wird, um die Textausgabe des Modells zu spezifizieren. Liang et al.[81] beobachteten, dass bereits simple *Prompts* die Detektionsrate von Texten, die von *ChatGPT* generiert wurden, erheblich verringerten. Konträr dazu ergaben die Studien von *Cai und Cui*[82] abweichende Resultate. Ihr *Prompt*, der das Simulieren menschlichen Verhaltens zum Ziel hatte, resultierte in einer erhöhten Identifikation der Texte als *KI*-generiert. Des Weiteren waren Stilmodifikationen durch das *Prompt Engineering*, wie beispielsweise „Answer the question in slang style.“[83], nicht zielführend. Es wäre erforderlich, intensivere *Slangs* oder stärkere Stilwechsel zu implementieren, um eine erfolgreiche Umgehung der Detektion zu erreichen. Doch in praktischen Anwendungen wäre dies nicht realisierbar, da derartige Stilistiken in realen Kontexten unbrauchbar wären.

Spoofing-Angriffe: Ein robuster Detektor sollte in der Lage sein, *KI*-generierte Texte zuverlässig als solche zu identifizieren, während er die Falsch-Positiv-Rate minimiert. Das bedeutet, er sollte vermeiden, menschlich erzeugte Texte fälschlicherweise als *KI*-generiert zu klassifizieren. Vor diesem Hintergrund

[76] Krishna u. a.: Paraphrasing, 2023, S. 4ff.
[77] Mitchell u. a.: Detectgpt, 2023.
[78] Tian: GPTZero, 2023.
[79] OpenAI: Classifier, 2023.
[80] Mitrović / Andreoletti / Ayoub: Chatgpt, 2023, S. 6.
[81] Liang u. a.: Detectors, 2023, S. 2.
[82] Cai / Cui: Evade, 2023, S. 5.
[83] Cai / Cui: Evade, 2023, S. 3.

identifizieren *Sadasivan et al.*[84] den *Spoofing*-Angriff als eine besondere Herausforderung für die Detektionsmechanismen. In diesem Angriffsszenario versucht der Angreifer, einen von einem Menschen verfassten Text so umzugestalten, dass er vom Detektor als *KI*-generiert eingestuft wird. Das Ziel solcher Angriffe besteht darin, die Vertrauenswürdigkeit des Detektors zu untergraben.

Syntaxänderungen: Unterschiede in der Syntax, die auf den ersten Blick nicht bemerkt werden und die Semantik des Satzes nicht ändern, können erhebliche Auswirkungen auf die Textklassifizierung haben.

Cai und Cui[85] stellten fest, dass die Leistungsfähigkeit von *SOTA*-Detektoren für *ChatGPT* signifikant abnimmt, wenn ein Leerzeichen vor einem beliebigen Komma im Text eingefügt wird. Einige Detektoren wurden durch diese Methode fast zu 100% getäuscht.

In Ergänzung dazu demonstrierten *Wolff* und *Wolff*[86] die erfolgreiche Umgehung von Textdetektoren, indem sie ausgewählte Zeichen durch visuell ähnliche Homoglyphen ersetzten.

Die Bekämpfung dieser Angriffsformen erweist sich als besonders relevant, da sie einfach durchzuführen sind und die Qualität des Textes nicht beeinträchtigen. Diese Änderungen sind so geringfügig, dass sie von einem menschlichen Betrachter normalerweise nicht bemerkt werden würden.[87]

Variation von Parametern bei der Textgenerierung: Die *SOTA*-Detektoren werden häufig nur auf *KI*-generierten Texten trainiert, die durch ein einziges *LLM* unter festen Parametern erstellt wurden.[88] Mit dem Aufkommen mehrerer *SOTA-LLMs*, wie *ChatGPT*[89], *Claude 2*[90], *LLaMA 2*[91] und *Bard*[92], können Angreifer das *LLM* auswählen, das die Detektoren am effektivsten umgeht. Zudem können sie die Parameter während des Generierungsprozesses variieren. *Pu et al.*[93] modifizierten beispielsweise den *Top-k*, *Top-p* und

[84] Sadasivan u. a.: ai-generated, 2023, S. 17ff.
[85] Cai / Cui: Evade, 2023, S. 5.
[86] Wolff / Wolff: Attacking, 2020, S. 2ff.
[87] Cai / Cui: Evade, 2023, S. 3.
[88] Pu u. a.: Deepfake, 2022, S. 8.
[89] OpenAI: ChatGPT, 2022.
[90] Anthropic: Claude, 2023.
[91] Touvron u. a.: LLaMA, 2023.
[92] Pichai: Next, 2023.
[93] Pu u. a.: Deepfake, 2022, S. 9f.

die *Temperatur* des generierenden *LLMs* und stellten fest, dass schon diese einfachen Anpassungen ausreichten, um die Detektoren effektiv zu umgehen. Der Rückgang des *Recalls* der Detektoren lag zwischen 9,7% bis 97,6%.

2.3.2. Gegenmaßnahmen

In ihrem 2022 veröffentlichten Artikel untersuchten *Pu et al.*[94] die Effizienz von *SOTA*-Detektoren im Hinblick auf adaptive Angriffe. Diese Untersuchung ist essenziell, um die Eignung dieser Detektoren für reale Anwendungen zu gewährleisten. Die Ergebnisse von *Pu et al.* deuten darauf hin, dass viele dieser Detektoren für Angriffe anfällig sind und über keine effektiven Abwehrmechanismen verfügen.

Die Forschung zur Integration von Abwehrmechanismen für die robuste Erkennung von *KI*-generierten Texten steht noch am Anfang. In diesem Zusammenhang werden in diesem Kapitel allgemeine Gegenmaßnahmen aus dem Bereich der natürlichen Sprachverarbeitung diskutiert. Drei Methoden scheinen dabei besonders vielversprechend für die Entwicklung eines robusten Detektors zu sein.

Adversarial Training

Goodfellow et al.[95] stellten erstmals den Ansatz des *Adversarial Trainings* vor. Beim *Adversarial Training* werden *Adversarial Examples* in den Trainingsdaten verwendet. Durch die Integration dieser Daten wird die Robustheit des Modells gegenüber diesen *Adversarial Examples* erhöht. Dabei beeinflusst die Stärke dieser Beispiele die finale Robustheit des Modells.

Ein Limit dieser Methode besteht jedoch darin, dass das Modell schnell überangepasst sein kann und dann Schwierigkeiten hat, auch nur leicht modifizierte *Adversarial Examples* zu erkennen, die es während des Trainings nicht gesehen hat.[96] Darüber hinaus wird vorausgesetzt, dass die genauen Details

[94] Pu u. a.: Deepfake, 2022, S. 7ff.
[95] Goodfellow / Shlens / Szegedy: Adversarial, 2014, S. 1.
[96] Jia / Liang: Adversarial, 2017, S. 8.

der Angriffsstrategie bekannt sind und eine ausreichende Menge an *Adversarial Examples* für das Training vorhanden ist. Daher ist das *Adversarial Training* begrenzt darin, unbekannte *Adversarial Attacks* abzuwehren.[97]

Korrektur von Perturbationen

Perturbationen bezeichnen in diesem Kontext manipulative Veränderungen oder Störungen der Eingabedaten. Dies kann beispielsweise das Einfügen oder Löschen von Zeichen eines *Tokens* beinhalten. Solche Veränderungen führen zur Entstehung von *Adversarial Examples*.[98]

Um die Anfälligkeit von Modellen gegenüber Manipulationen zu verringern, kann die Korrektur der Perturbationen eingesetzt werden. Hierbei wird häufig auf Rechtschreib- und Grammatikkorrekturen zurückgegriffen. Sowohl *Liu et al.*[99] als auch *Gao et al.*[100] entwickelten Modelle, die auf der Idee beruhen, fehlerbehaftete Wörter in den Eingaben zu erkennen und zu korrigieren.

Liu et al. entwarfen ein zweistufiges Rechtschreibkorrekturmodell, das fehlerhafte Wörter erkennt und korrigiert. Bei der Anwendung auf Filmbewertungen stellten sie fest, dass die Genauigkeit der Sentimentanalyse bei Texten mit Rechtschreibfehlern durch die Nutzung ihres Korrekturmodells deutlich verbessert wurde. Ihre Ergebnisse zeigten, dass dieser Abwehrmechanismus sogar signifikant effektiver war als das *Adversarial Training*[101].

In ähnlicher Weise konnten *Gao et al.*[102] die Vorhersagegenauigkeit von Modellen mithilfe eines Autokorrektors bei Texten mit kleineren Modifikationen, wie das Ändern, Löschen oder Tauschen eines Zeichens, verbessern. Allerdings stellten sie fest, dass die Korrektur nicht gegen eine Modifikation von mehreren Zeichen robust ist. Darüber hinaus benötigt ein Autokorrektor zusätzliche Ressourcen, die den Trainingsprozess des *NLP*-Modells komplexer gestalten können.

[97] Li u. a.: Textbugger, 2018, S. 13f.
[98] Goodfellow / Shlens / Szegedy: Adversarial, 2014, S. 1.
[99] Liu u. a.: Spelling, 2020, S. 556f.
[100] Gao u. a.: Evade, 2018, S. 12.
[101] Liu u. a.: Spelling, 2020, S. 558.
[102] Gao u. a.: Evade, 2018, S. 12.

Robustheit durch Zertifizierung

Während die bisher vorgestellten Abwehrmechanismen sich hauptsächlich auf den Schutz gegen bekannte Textmanipulationen konzentrieren, bietet die Robustheit durch Zertifizierung einen umfassenderen Schutzansatz. Der Ansatz versucht dem Modell eine obere Grenze für den schlimmsten Fall von Manipulationen zu setzen. Damit kann die Zertifizierung garantieren, wie das Modell unter neuartigen oder zuvor ungesehenen Angriffen reagieren wird.[103]

In diesem Kontext haben *Zeng et al.*[104] eine Methode namens *Randomized Mask* vorgestellt, die sich in ihren Experimenten als robust gegenüber einer Vielzahl von Perturbationen auf Wort- und Zeichenebene erwiesen hat. Die Kernidee hinter *Randomized Mask* ist die zufällige Maskierung einer ausreichenden Anzahl von *Tokens* in einem gegebenen Text. Durch die Maskierung wird es für Angreifer schwieriger, vorherzusagen, welche Teile des Textes für die Modellvorhersage entscheidend sind. Abbildung 2.5 illustriert diesen Prozess detailliert. Hierbei repräsentiert *[MASK]* die Maskierungen, die von *Randomized Mask* vorgenommen werden. Durch wiederholtes Anwenden dieser Maskierungen auf den Eingabetext entstehen mehrere Varianten, welche dann einzeln klassifiziert werden, um daraus eine Klassifikation für den gesamten Eingabetext abzuleiten.

Die zugrunde liegende Annahme ist, dass durch diese Maskierung die Wahrscheinlichkeit drastisch reduziert wird, dass alle vom Angreifer modifizierten Wörter in den maskierten Kopien präsent sind. In den Experimenten von *Zeng et al.*[105] hat *Randomized Mask* die Klassifikation von über 50% der Sätze im *AGNEWS*-Datensatz gegen Perturbationen von bis zu fünf Wörtern robust gemacht.

[103] Goyal u. a.: Survey, 2022.
[104] Zeng u. a.: Mask, 2021, S. 2f.
[105] Zeng u. a.: Mask, 2021, S. 1.
[106] Zeng u. a.: Mask, 2021, S. 2

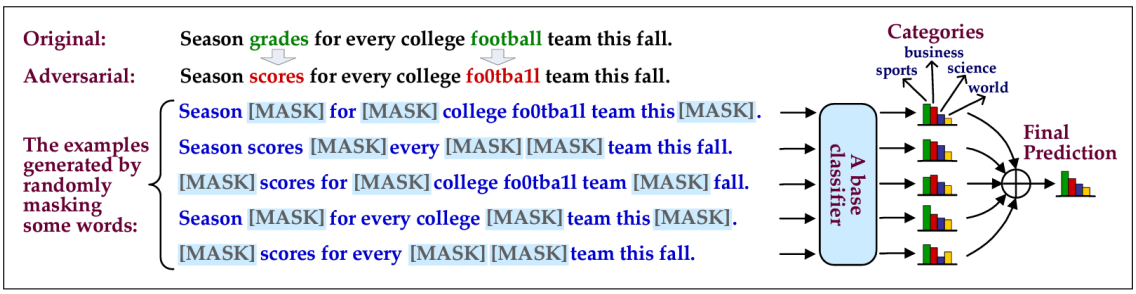

Abbildung 2.5.: Klassifikation durch *Randomized Masks*.[106]

2.4. Limitationen und Erkenntnisse der bestehenden Forschung

Die Auseinandersetzungen mit dem momentanen Stand der Forschung bei der Erkennung von *KI*-generierten Texten hat wichtige Erkenntnisse geliefert, aber auch vorhandene Limitationen aufgezeigt, die weitere Forschungsarbeit benötigen.

Ein Vergleich der *Zero-Shot* Methode mit einer Klassifikation, die durch *Fine-Tuning* von *RoBERTa* durchgeführt wurde, ergab, dass der letztgenannte Ansatz einen überlegenen Detektor darstellt. Dieser zeichnet sich durch eine höhere Genauigkeit aus und zeigte eine gesteigerte Effizienz in *OOD*-Szenarien.[107] Diese überlegene Performance der feinabgestimmten *LLMs* bei der Detektion zeigt sich auch im Vergleich mit einfachen und merkmalsbasierten Detektoren.[108]

Guo et al.[109] stellten fest, dass ein Detektor, der auf die Klassifikation individueller Sätze trainiert wurde, verbesserte Resultate bei der Identifikation von *KI*-generierten Texten liefert. Eine vorherige Filterung der Textdaten hatte in ihren Experimenten keinen signifikanten Einfluss auf die Ergebnisse. Laut *Solaiman et al.* tragen Trainingsdatensätze mit variablen Textlängen im Vergleich zu jenen mit konstanten Längen zu einer erhöhten Genauigkeit des Detektors bei.[110] Die Autoren erkannten zudem, dass vielfältige Datensätze die Generalisierungsfähigkeit der Detektoren verbessern. Zusätzlich erschweren umfangreichere Modelle in der Generation die Identifikation der künstlichen Texte.

[107] Guo u. a.: Close, 2023, S. 12f.
[108] Solaiman u. a.: Impacts, 2019, S. 12; Fröhling / Zubiaga: Feature, 2021, S. 16.
[109] Guo u. a.: Close, 2023, S. 14.
[110] Solaiman u. a.: Impacts, 2019, S. 15.

Die aktuellen Forschungsergebnisse weisen deutlich auf Limitationen in der Robustheit von Detektoren hin. Sowohl syntaktische Manipulationen, wie das Hinzufügen von Rechtschreibfehlern[111], zusätzlichen Leerzeichen[112] oder Homoglyphen[113], als auch semantische Manipulationen, etwa durch Umformulierungen der Texte[114], können die *SOTA*-Detektoren täuschen. Trotz dieser Erkenntnisse wurden in der Forschung bisher keine signifikanten Fortschritte im Hinblick auf die Abwehr dieser Angriffe erzielt. Darüber hinaus besteht in der wissenschaftlichen Gemeinschaft Uneinigkeit darüber, in welchem Maße die Gestaltung spezieller *Prompts* die Erkennung von *KI*-generierten Texten beeinflusst.[115]

Ein weiteres Defizit in der aktuellen Forschung liegt in der überwiegenden Fokussierung auf die Klassifikation von englischsprachigen Texten. Der von *Guo et al.*[116] entwickelte Detektor zur Klassifikation chinesischsprachiger Texte stellt hierbei eine Ausnahme dar.

[111] Gao u. a.: Evade, 2018, S. 10.
[112] Cai / Cui: Evade, 2023, S. 5.
[113] Wolff / Wolff: Attacking, 2020, S. 2ff.
[114] Krishna u. a.: Paraphrasing, 2023, S. 4ff.
[115] Liang u. a.: Detectors, 2023, S. 2; Cai / Cui: Evade, 2023, S. 3.
[116] Guo u. a.: Close, 2023, S. 12f.

3. Methodik

Die vorliegende Arbeit untersucht die binäre Klassifikation von Textdaten, wobei zwischen *KI*-generierten Texten und menschlich verfassten Texten unterschieden wird. Die Prognose stützt sich ausschließlich auf die als Eingabe bereitgestellten Textdaten. Das bedeutet, dass weder die für die Generierung verwendeten *Prompts* noch jegliche Meta-Informationen berücksichtigt werden. Die Beschränkung auf diese Informationen gewährleistet, dass der Ansatz hinsichtlich seiner Anwendbarkeit universell bleibt und reale Anwendungssituation möglichst genau widerspiegelt. Ziel dieser Forschung ist es, einen robusten Detektor zur Klassifizierung von *KI*-generierten Texten in deutscher Sprache zu entwickeln.

3.1. Daten

Der Prozess der Datenaufbereitung ist in Abbildung 3.1 visualisiert. Diesem Prozess wurde besondere Beachtung geschenkt, da aus vorangegangenen Studien hervorgeht, dass die Qualität des Trainingsdatensatzes einen signifikanten Einfluss auf die Effektivität des Detektors in realen Szenarien hat. Die in der Abbildung 3.1 präsentierten Schritte werden im Verlauf dieses Kapitels detailliert erläutert.

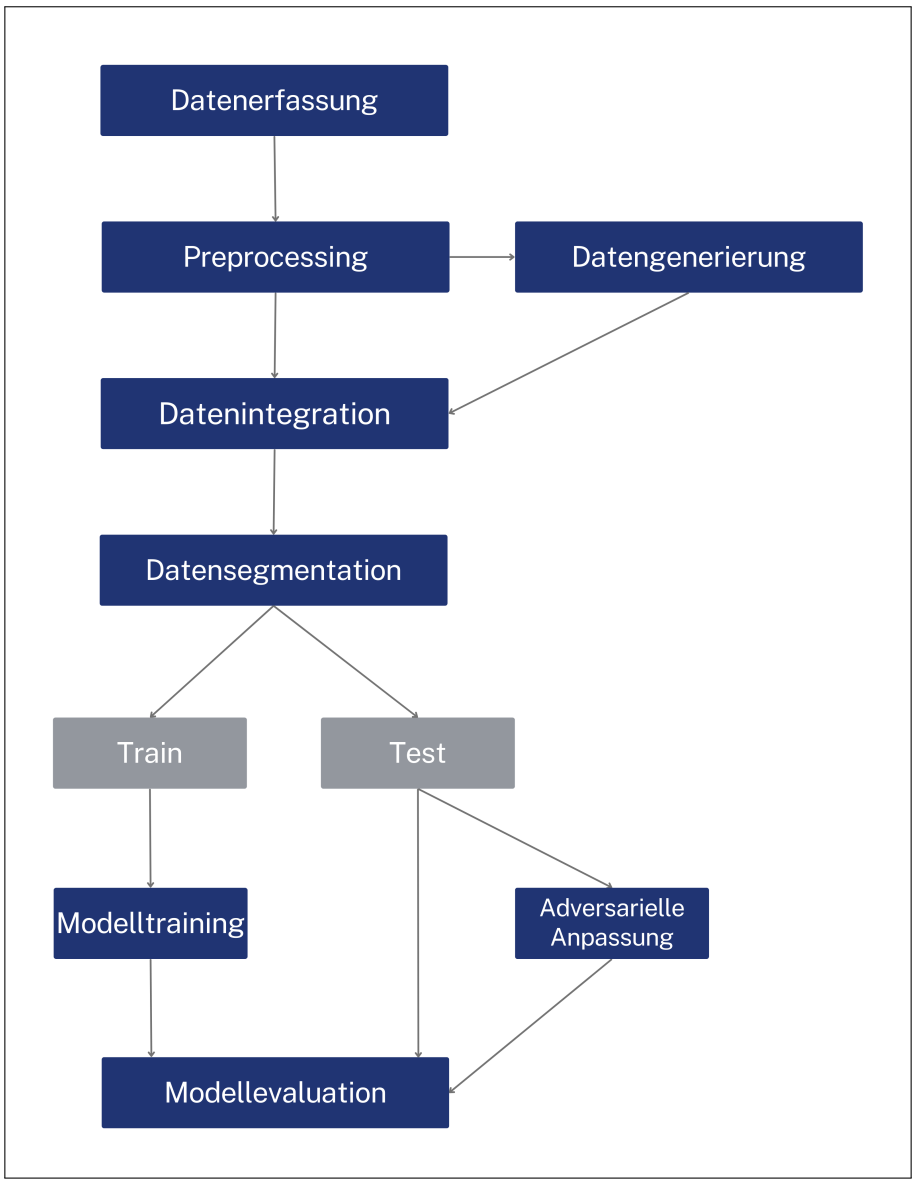

Abbildung 3.1.: Überblick über die Datenaufbereitung

3.1.1. Datenerfassung

Im Rahmen dieser Untersuchung wurde festgestellt, dass in der Literatur bisher kein Datensatz existiert, welcher ein solch vielfältiges Spektrum an Daten abdeckt. Die meisten existierenden Datensätze tendieren dazu, sich auf eine spezifische Textgattung aus einem Bereich zu konzentrieren. Für die vorliegende Studie wurde eine umfangreiche Sammlung menschlich verfasster Texte aus diversen Quellen konsolidiert. Bei dieser Sammlung wurden bestehende Datensätze integriert und zusätzlich spezifische Daten eigenhändig aus Datenbanken extrahiert. Für jeden Datensatz wurden menschlich

generierte Texte sowie Kontextinformationen, die für die anschließende Generierung der künstlichen Texte genutzt werden, berücksichtigt. Nachfolgend sind die vorhandenen Textgenres und die Quellen der Textdaten detailliert aufgelistet.

1. **Dienstleistungsbewertungen:** Dieser Datensatz[1] stammt von einer deutschen Online-Plattform, auf der Patienten ihre Ärzte bewerten können. Diese Bewertungen umfassen Mediziner aus allen Fachrichtungen. Aus diesem Datensatz wurden Bewertungstexte und schulnotenbasierte Bewertungen entnommen.

2. **Fachartikel:** In dieser Arbeit wurden Datenbanken von fünf Hochschulen durchsucht, um Titel und Zusammenfassungen wissenschaftlicher Arbeiten zu extrahieren. Dieses Spektrum umfasst studentische Arbeiten, wie Bachelor- und Masterarbeiten, Dissertationen sowie weitere Hochschulveröffentlichungen. Die Zusammenfassungen dienen als menschlich generierte Texte und die Titel als Kontextinformationen.

3. **Forenbeiträge:** Als Forenbeiträge werden Kommentare aus dem Internet-Forum *Reddit* verwendet. Aufgrund von *Application Programming Interface* (*API*)-Änderungen bei *Reddit* ist es nicht mehr möglich auf historische Daten zurückzugreifen. Deswegen ist es erforderlich, eine eigene Applikation zur Datengewinnung zu entwickeln. Diese Applikation zieht für die Datensammlung beliebte deutsche *Subreddits* heran. Bei der Extraktion der Beiträge werden solche ausgewählt, die weder Bilder noch Links enthalten. Zur Sicherstellung einer angemessen Qualität werden Beiträge anhand ihrer Upvote-Zahlen gefiltert. In dieser Studie fungieren die extrahierten Beitragsdaten als Kontextinformationen zur Textgenerierung, während die zugehörigen Kommentartexte als menschlich annotierte Textdaten herangezogen werden.

4. **Kinderlexikonartikel:** *Aumiller und Gertz*[2] stellen Artikel des Kinderlexikons *Klexikon* bereit. Im Gegensatz zu *Wikipedia*-Artikeln sind diese in vereinfachter Sprache verfasst und tendenziell kürzer. Aus diesen Datensatz werden sowohl die Textinhalte der Kinderlexikonartikel als auch die zugehörigen Kontextstrukturen, repräsentiert durch die jeweiligen Kapitel, entnommen.

[1] o.V.: Doctors, 2021.
[2] Aumiller / Gertz: Klexikon, 2022.

5. **Lexikonartikel:** Ebenfalls aus dem Datensatz von *Aumiller und Gertz* werden Artikel der Online-Enzyklopädie *Wikipedia* entnommen. Parallel dazu wird der von *Deepset* entwickelte Frage-Antwort-Datensatz *GermanQuAD*[3] herangezogen, welcher ebenfalls auf *Wikipedia*-Einträgen basiert. Ähnlich wie bei den Kinderlexikonartikeln werden auch in diesen beiden Datensätzen Kapitel- und Unterkapitelstrukturen als Kontextinformationen genutzt.

6. **Produktbewertungen:** Aus dem multilingualen Datensatz von *Keung et al.*[4] werden die deutschen Produktbewertungen von Kunden auf dem *Amazon*-Marktplatz für Produkte diverser Kategorien berücksichtigt. Die Bewertungstexte repräsentieren die menschlich verfassten Texte, während der Titel, die Sternebewertung und die Produktkategorie als Kontextinformationen herangezogen werden.

7. **Tweets:** *Kratzke*[5] präsentiert Auszüge von deutschen *Tweets*, die im Zeitraum von April 2019 bis April 2020 veröffentlicht wurden. Die in den *Tweets* enthaltenen *Hashtags* dienen als Kontextinformationen.

8. **Zeitungsartikel:** *Mi* (2020)[6] sammelte Artikel von 48 Zeitungen, darunter *Bild, FAZ, Focus, Spiegel* und *Welt*. Die aus diesen Artikeln extrahierten Titel und Texte dienen in dieser Arbeit als Kontextinformationen und Datenbasis für das Training und die Evaluation.

Es wird sichergestellt, dass die Texte idealerweise vor dem November 2022 erstellt wurden, um den Einfluss der öffentlichen Verfügbarkeit von *LLMs* auf die Texte zu minimieren. Bei der Datenerhebung wird stets versucht, vorhandene deutschsprachige Datensätze als erste Anlaufstelle zu nutzen. Falls solche nicht verfügbar sind, werden die Daten eigenständig, analog zu den Forenbeiträgen und Fachartikeln, gesammelt.

[3] Möller / Risch / Pietsch: GermanQuAD, 2021.
[4] Keung u. a.: Reviews, 2020.
[5] Kratzke: Tweets, 2020.
[6] Mi: News, 2020.

3.1.2. Preprocessing

Jeder Datensatz wird sorgfältig daraufhin überprüft, ob die enthaltenen Texte ausschließlich in deutscher Sprache verfasst sind und ob leere oder redundante Zeilen vorhanden sind. Texte mit einer untypisch geringen Zeichenanzahl werden ausgeschlossen, da solche Einträge oft auf Fehler während des *Crawling*-Prozesses zurückzuführen sind. Sowohl bei *KI*-generierten als auch bei menschlich erstellten Texten wird eine Beschränkung auf eine maximale Zeichenlänge von 3200 Zeichen beziehungsweise 700 Tokens vorgenommen, um sicherzustellen, dass die maximale Kontextlänge der feinabgestimmten *LLMs* nicht überschritten wird. Insbesondere werden für jeden der Datensätze spezifische Bereinigungsprozeduren angewandt. Der zugehörige Programmcode kann auf der beigelegten CD eingesehen werden.

Es wird darauf geachtet, dass die einzelnen Datenquellen eine hohe interne Vielfalt aufweisen, um eine repräsentative Auswahl sicherzustellen. Beispielsweise wird bei den Zeitungsartikeln eine Obergrenze von 500 Artikeln pro Zeitung für die Einbeziehung in dem Datensatz festgelegt. Darüber hinaus werden Emojis aus den Texten entfernt, um die Analyse auf reine Textdaten zu beschränken und einen potenziellen *Bias* durch bestimmte Emojis zu vermeiden.

Aufgrund der großen Datenmenge in den Ausgangsdaten wird eine Zufallsstichprobe angewendet, die die Datenmenge reduziert. Abschließend werden die verarbeiteten Daten aufgrund der Kompaktheit und einfachen Handhabung in einer *CSV*-Datei gespeichert.

3.1.3. Generierung

Die Generierung künstlicher Textdaten stellt einen zentralen Aspekt dieser Arbeit dar. Die vier prominentesten Modelle für die Generierung von deutschsprachigen Texten sind *ChatGPT*[7] von *OpenAI*, *Bard*[8] von *Google*, *Claude2*[9] von *Anthropic* und *LLaMA 2* von *Meta*.

[7] OpenAI: ChatGPT, 2022.
[8] Pichai: Next, 2023.
[9] Anthropic: Claude, 2023.

Es sei angemerkt, dass nicht alle dieser Modelle in der vorliegenden Arbeit aufgrund von Zugangsbeschränkungen berücksichtigt werden.

Closed-Source Modelle

ChatGPT[10]: *OpenAI* hat am 30. November 2022 *ChatGPT* für die allgemeine Öffentlichkeit zugänglich gemacht. Die erste Version von *ChatGPT* basiert auf der *GPT-3.5* Architektur. Seit dem 13. März 2023 können Abonnenten von *ChatGPT Plus* auf das erweiterte Modell *GPT-4*[11] zugreifen. *OpenAI* stellt eine *API* bereit, über die Nutzer auf Basis der verbrauchten *Tokens* abgerechnet werden. Das *GPT-3.5 Turbo* Modell wird von *OpenAI* als eine kosteneffiziente Alternative präsentiert, die nur geringfügig weniger leistungsfähig als das neueste Modell, *GPT-4*, ist. Deswegen wird in dieser Untersuchung *GPT-3.5 Turbo* zur Generierung der künstlichen Texte genutzt.

Claude2[12]: *Claude 2*[13] ist seit dem 11. Juli 2023 öffentlich zugänglich. Paradoxerweise produziert dieses Modell zwar deutsche Texte, aber kann nur über Umwege eines *Virtual Private Networks (VPN)* in Deutschland genutzt werden. Obwohl es keine offizielle *API* von *Anthropic* für dieses Modell gibt, existieren inoffizielle *APIs* auf *GitHub*, die mithilfe von *Cookies* eigenständige Chats mit dem Modell erzeugen können. Unglücklicherweise beschränkt *Anthropic* die Anfragen über solche Schnittstelle, sodass eine konstante Nutzung nicht möglich ist. Infolgedessen wird *Claude 2* nicht für die Generierung von Trainings- und Testdaten in dieser Arbeit herangezogen.

Bard[14]: *Google* hat *Bard* am 6. Februar 2023 in englischer Sprache eingeführt und die deutsche Version folgte am 13. Juli 2023.[15] Dieses Modell weist ähnliche Einschränkungen wie *Claude 2* auf und wird dementsprechend nicht für die Datenproduktion in Betracht gezogen.

[10] OpenAI: ChatGPT, 2022.
[11] OpenAI: GPT-4, 2023.
[12] Anthropic: Claude, 2023.
[13] Anthropic: Claude, 2023.
[14] Pichai: Next, 2023.
[15] Krawczyk: Deutschland, 2023.

Open-Source Modelle

Durch die Veröffentlichung der *LLaMA 2*-Modelle[16] mit öffentlich zugänglichem Quellcode haben sich die *Open-Source* Sprachmodelle enorm entwickelt. Jedoch liegt der Fokus dieser Modelle aufgrund ihres primären Trainingsdatensatzes überwiegend auf der englischen Sprache, was dazu führt, dass hauptsächlich englischsprachige *Open-Source LLMs* publiziert werden. Die derzeit verfügbaren deutschsprachigen *Open-Source* Sprachmodelle können Texte nicht in einer Qualität generieren, die mit der von aktuellen *SOTA-LLMs* renommierter Technologieunternehmen vergleichbar ist. Zudem stellt die notwendige *Hardware*-Infrastruktur für den Betrieb dieser Modelle eine signifikante Hürde dar. Aufgrund dieser Limitationen wird in der vorliegenden Masterarbeit auf den Einsatz von *Open-Source* Modellen für die Textgenerierung verzichtet.

Generierungsprozess

Für die Generierung von Texten werden aus den vorab gesammelten Datensätzen relevante Kontextinformationen extrahiert und dem *GPT-3.5*-Modell in Form von *Prompts* zugeführt. Diese Kontextdaten werden in natürliche Sprache übersetzt und dem Modell als zusammenhängender Fließtext übermittelt. Um eine klare Abgrenzung zwischen Instruktionen und Informationen aus den Daten zu gewährleisten, werden die Kontextinformationen mit Anführungszeichen hervorgehoben.

Für jede der acht Textarten wird ein individuelles *Prompt-Set* erstellt, das aus sieben unterschiedlichen *Promptvorlagen* besteht. Tabelle 3.1 präsentiert beispielhaft die *Promptvorlagen* für die Generierung von *Tweets*. Zunächst wird der *Standardprompt A* für Tweets erstellt, welcher lediglich die notwendigen Informationen enthält, um künstliche *Tweets* anhand von Kontextinformationen in Form von *Hashtags* zu generieren. Während des Generierungsprozesses hat es sich als vorteilhaft erwiesen, dem Modell explizit mitzuteilen, ausschließlich den Kommentartext auszugeben. Dies vermeidet, dass Formulierungen wie „Natürlich, hier ist ein Vorschlag.“[17] oder „Als

[16] Touvron u. a.: LLaMA, 2023.
[17] OpenAI: ChatGPT, 2022.

KI-Sprachmodell ...“[18] im generierten Text erscheinen. Die *Prompts B* und *C* zielen darauf ab, die Textlänge der Ausgaben von *GPT-3.5* zu regulieren, um eine Länge zu erzielen, die den menschlichen Texten ähnlicher ist. Üblicherweise neigt *GPT-3.5* dazu, längere Texte zu produzieren, besonders bei kurzen Textformen wie *Tweets*. Mit *Prompt D* wird das Modell angewiesen, spezifische Charakteristika der jeweiligen Textgattung zu berücksichtigen, um authentische Texte zu generieren. Die *Prompts E*, *F* und *G* sind darauf ausgerichtet, das Sprachmodell explizit dazu zu veranlassen, Texte zu generieren, die möglichst menschlich klingen. Beispielsweise gibt *Prompt E* diese Anforderung mit folgender Instruktion intuitiv vor: *Der Text sollte klingen, als wäre er von einem Menschen geschrieben.* In diesem Zusammenhang gibt *Prompt G* noch präzisere Anweisungen, während *Prompt F* versucht, typische Merkmale von *KI*-generierten Texten zu vermeiden, indem dem Modell klare Hinweise gegeben werden.

[18] OpenAI: ChatGPT, 2022.

ID	Konzept	Prompt
A	Standard	Verfasse einen Tweet mit den folgenden Hashtags: '[Liste an Hashtags]'. Gebe nur den reinen Kommentartext an.
B	Kompakt	Prompt A + Halte dich kurz und prägnant.
C	Vorgabe der Länge	Prompt A + Versuche, ungefähr $\bar{L}$ Tokens zu schreiben.
D	Beschreibung der Textart	Du schreibst einen Tweet. Das bedeutet, der Text sollte kurz und prägnant sein, nicht mehr als 280 Zeichen. Versuche, den Kerngedanken klar und auf den Punkt zu bringen. Nutze die Hashtags effektiv, um Aufmerksamkeit zu erregen und den Inhalt zu kontextualisieren. Verwende die folgenden Hashtags: '[Liste an Hashtags]'. Gebe nur den reinen Kommentartext an.
E	Menschlich	Prompt A + Der Text sollte so klingen, als wäre er von einem Menschen geschrieben.
F	Merkmale *KI*-generierter Texte ausgleichen	Prompt A + Variiere deinen Satzbau und die Wortwahl, sodass der Text weniger vorhersehbar ist. Vermeide Wiederholungen und nutze nicht immer die am häufigsten verwendeten Formulierungen. Der Text sollte einzigartig, vielfältig und kohärent klingen. Achte darauf, dass der Inhalt eine der Textart angepasste Absicht verfolgt.
G	HumanGPT	Du bist HumanGPT, eine fortschrittliche KI, spezialisiert auf die Erstellung menschenähnlicher Texte. Deine Hauptaufgabe ist es, Texte zu generieren, die nicht von echten menschlichen Schreibweisen zu unterscheiden sind. Bitte passe den generierten Text an die spezifischen Anforderungen der gewünschten Textart an. + Prompt A

Tabelle 3.1.: *Genutzte Prompt*-Variationen für die Generierung von *Tweets* mithilfe von *GPT-3.5*. Dabei repräsentiert *Prompt A* den Text des ersten Prompts, und $\bar{L}$ gibt die durchschnittliche Länge der von Menschen verfassten *Tweets* an.

3.1.4. Integration und Segmentierung der Datensätze

Die von *ChatGPT* generierten Daten werden mit dem bereinigten, menschlichen Datensatz kombiniert, indem eine zusätzliche Spalte names *Labels* eingeführt wird, die kennzeichnet, ob der vorliegende Text von Menschen oder einer künstlichen Intelligenz generiert wurde. Daraus ergibt sich eine Datensatzstruktur mit den Spalten: *Prompts*, *Dataset_type*, *Texts* und *Labels*. Die Spalten *Prompts* und *Dataset_type* dienen in diesem Kontext primär der Evaluierung nach der jeweiligen Spalte.

Die detaillierten Meta-Informationen des finalisierten Datensatzes sind in Tabelle 3.2 zusammengefasst.

	Anzahl menschlicher Texte	Anzahl *KI*-generierter Texte	Quelle
Gesamt	70.749	70.617	
Dienstleistungsbew.	10.000	9.980	Bewertungsportal [19]
Fachartikel	8.839	8.821	Online-Bibliotheken
Forenbeiträge	8080	8.069	*Reddit*
Kinderlexikonartikel	5.609	5.597	*Klexikon* [20]
Lexikonartikel	7.950	7.924	*Wikipedia* [21]
Produktbewertungen	10.000	9.991	*Amazon* [22]
Tweets	10.000	9.994	*Twitter* [23]
Zeitungsartikel	10.271	10.241	48 Online-Zeitungen [24]

Tabelle 3.2.: Übersicht über die Meta-Informationen des erstellten deutschen Datensatzes. Dieser besteht aus 70.749 menschlich verfassten Texten und 70.617 *KI*-generierten Texten, verteilt auf acht verschiedene Textarten.

Für die nachfolgende Analyse wird der kombinierte Datensatz in Trainings- und Testdatensatz im Verhältnis 80:20 unterteilt. Während des Teilungsprozesses wurde darauf geachtet, dass sowohl die Verteilung der verschiedenen *Prompts* als auch das Verhältnis von menschlich generierten zu *KI*-generierten Daten in beiden Teildatensätzen gleich bleibt.

Die Entscheidung für diese spezifische Aufteilung basiert auf den gängigen Praktiken im maschinellen Lernen, welche darauf abzielen, ein ausgewogenes Verhältnis zwischen Trainings- und Testdaten zu gewährleisten und damit eine robuste Evaluierung des Modells zu ermöglichen.

[19] o.V.: Doctors, 2021
[20] Aumiller / Gertz: Klexikon, 2022
[21] Möller / Risch / Pietsch: GermanQuAD, 2021; Aumiller / Gertz: Klexikon, 2022
[22] Keung u. a.: Reviews, 2020
[23] Kratzke: Tweets, 2020
[24] Mi: News, 2020

3.1.5. Adversarial Attacks

Um die Robustheit des Modells unter realen Bedingungen zu evaluieren, wurden *Adversarial Examples* in das Experiment integriert. Diese *Adversarial Examples* zielen in dieser Untersuchung speziell darauf ab, dass *KI*-generierte Texte als menschliche Texte klassifiziert werden. Die *Adversarial Attacks* werden ausschließlich auf den *KI*-generierten Texten innerhalb des Testdatensatzes angewendet. Im Folgenden werden die verwendeten *Adversarial Attacks* dargestellt:

Homoglyphen

In dieser Arbeit werden sämtliche identifizierbaren Homoglyphen aus der *Confusable Homoglyphs* Bibliothek entnommen. Anschließend werden die optimalen kyrillischen und griechischen Homoglyphen ausgewählt, die visuell ähnliche Buchstaben des im Deutschen verwendeten lateinischen Alphabets ersetzen. Die genaue Liste der ausgewählten Homoglyphen ist im Anhang A.1 dargestellt.

Leerzeicheninsertion

Für das zufällige Hinzufügen von Leerzeichen wurde ein Algorithmus entwickelt, der vor beliebigen Kommata ein zusätzliches Leerzeichen einfügt. Dies wird vor zwei Kommata in jedem *KI*-generierten Text durchgeführt.

Rechtschreibfehler

Zur Implementierung dieses Angriffes wird ein Wörterbuch erstellt, das korrekten deutschen Schreibweisen ihre geläufigen fehlerhaften Versionen gegenüberstellt. Die exemplarische Liste der häufig falsch geschriebenen Wörter wird mit *ChatGPT* generiert.

Tippfehler

Tippfehler, die gewöhnlicherweise aus unbeabsichtigten Fehlern bei der Eingabe resultieren, werden durch die *NLPAug*-Bibliothek in Python emuliert. Mit Hilfe der *KeyboardAug*-Klasse werden gezielt Tippfehler reproduziert, wobei 15 Prozent der Zeichen in zehn Prozent der Wörter vertauscht werden. Im Vorfeld wurden sämtliche Stoppwörter entfernt, um die Effektivität des Angriffes zu steigern.

Synonyme

Für diese *Adversarial Attack* wird *OdeNet* herangezogen, eine Datenbank, die speziell für die deutsche Sprache entwickelt wurde, um entsprechende Synonyme für die ausgewählten Wörter bereitzustellen. Nach der Filterung der Stoppwörter werden zehn Prozent der Wörter durch die Synonyme ersetzt. Ein zu beachtender Aspekt dieser Vorgehensweise ist, dass der Kontext der Wörter nicht berücksichtigt wird.

Paraphrasierung

Leider gibt es für diese *NLP*-Aufgabe kein zufriedenstellendes deutsches Sprachmodell, weswegen die Methode der Rückübersetzung zur Erstellung von Paraphrasen des Ausgangstextes genutzt wird. In dieser Arbeit wird der Text zuerst in die Zwischensprache Englisch übersetzt, um diesen dann wieder in die deutsche Ausgangssprache zu übersetzen. Mit Hilfe der *Translators*-Bibliothek wird zur Übersetzung der Texte auf die *API* des *Google Übersetzers* zugegriffen.

3.2. Architektur des Detektors

In allen nachfolgenden Experimenten werden feinabgestimmte *LLMs* zur Klassifizierung der *KI*-generierten Texte genutzt, da ihre Performance, wie in Kapitel 2.3 dargelegt, andere Detektionsmethoden signifikant übertrifft. Insbesondere *Encoder-Only* Modelle wie *RoBERTA* und *BERT* zeigen bemerkenswerte Ergebnisse bei der Identifikation von *KI*-generierten Texten.

Zur Realisierung der Klassifizierung durch *Encoder-Only Transformer* wird auf die Klasse *AutoModelForSequenceClassification* zurückgegriffen. Diese Klasse ist für verschiedene vortrainierte *Encoder-Only Transformer* wie *BERT*, *RoBERTa* und *XLM*[25] implementiert. Mit Hilfe einer *Pooling*-Schicht werden die Ausgaben des vortrainierten Modells auf einen Vektor konsolidiert. Anschließend wird eine lineare Schicht hinzugefügt, die den Vektor der *Pooling*-Schicht als Eingabe verwendet und das Klassifikationsergebnis liefert. Während des *Fine-Tunings* werden sowohl die Gewichte des vortrainierten Modells als auch der zusätzlichen Klassifikationsschicht angepasst. Da für die Klassifikation lediglich eine Schicht hinzugefügt wird, muss besonders darauf geachtete werden, dass eine Überanpassung verhindert wird. Die *Tokenisierung* wird durch den entsprechenden *Tokenizer* realisiert, der mittels der *AutoTokenizer*-Klasse aus der *Transformers*-Bibliothek geladen wird.

Für die Untersuchung werden verschiedene *Encoder-Only LLMs* verwendet, die speziell auf die Verarbeitung deutscher Texte angepasst wurden. Der erforderliche *Tokenizer* des Modells muss ein deutsches Vokabular besitzen, um eine korrekte Textverarbeitung zu gewährleisten. Da die genutzten Modelle auf *BERT* und *RoBERTa* basieren, liegt die maximale *Token*-Grenze der Eingabetexte bei 512.

Neueste Forschungsergebnisse von *Guo et al.*[26] empfehlen, Texte im Trainingsprozess in einzelne Sätze zu zerlegen und separat zu klassifizieren, um die Robustheit des Detektors zu steigern. Zudem zeigten ihre Untersuchungen, dass unveränderte Rohdaten die besten Resultate erzielen. Deshalb wird in der vorliegenden Arbeit lediglich eine vorgeschaltete Emoji-Filterung eingesetzt, um potenzielle Klassifikationsverzerrungen zu minimieren.

[25] Lample / Conneau: Cross, 2019.
[26] Guo u. a.: Close, 2023, S. 13f.

Obwohl das Training satzweise durchgeführt wird, wird bei der Inferenz der Text als Ganzes betrachtet.

Zusammenfassend basiert das in dieser Arbeit verwendete Basismodell auf einer Architektur, die in Einklang mit den aktuellen *SOTA*-Modellen steht, wie sie beispielsweise von *Guo et al.*[27] und *Solaiman et al.*[28] beschrieben werden.

3.3. Implementierung der Gegenmaßnahmen

In diesem Kapitel wird die Implementierung der Gegenmaßnahmen gegen *Adversarial Attacks* erläutert. Es wird darauf geachtet einen allgemeinen Anwendungsfall zu berücksichtigen, welcher nicht auf die spezifischen Merkmale der Attacken überangepasst ist. Im theoretischen Teil der Arbeit wurden drei zentrale Kategorien von Gegenmaßnahmen identifiziert: *Adversarial Training*, Korrektur von Perturbationen und Robustheit durch Zertifizierung.

3.3.1. Adversarial Training

Im *Adversarial Training* wird das Modell nicht nur mit dem herkömmlichen Trainingsdatensatz, sondern auch mit *Adversarial Examples* trainiert. Die *Adversarial Examples* werden durch Anwendung der im theoretischen Teil beschriebenen Attacken auf Teile des Trainingsdatensatzes generiert. Dabei wird sorgfältig darauf geachtet, dass sich die *Adversarial Examples* des Trainingsdatensatzes nicht mit denen des Testdatensatzes überschneiden.

3.3.2. Korrektur von Perturbationen

Im Rahmen dieser Arbeit werden zwei Korrekturmethoden vorgestellt, die den Ursprungstext ohne die Einflüsse von Perturbationen wiederherstellen.

[27] Guo u. a.: Close, 2023, S. 10f.
[28] Solaiman u. a.: Impacts, 2019, S. 13ff.

Für die Erkennung von Homoglyphen in den Eingabetexten wird die Bibliothek *Confusable Homoglyphs*[29] herangezogen. Die Effektivität dieses Ansatzes ist davon abhängig, ob die von einem potenziellen Angreifer genutzten Homoglyphen in der genannten Bibliothek enthalten sind. Nach ihrer Erkennung werden diese Homoglyphen in ihre korrespondierenden gemäß *American Standard Code for Information Interchange* (*ASCII*) kodierten Buchstaben zurückübersetzt.

Zusätzlich wird ein Autokorrektor eingesetzt, welcher Tipp-, Rechtschreib- und stilistische Fehler in dem Text erkennt und korrigiert. Rechtschreibkorrektoren wie *Symspell*[30] und *PySpellchecker*[31] wurden verworfen, da diese nur die Rechtschreib- und Tippfehler verbessern. Das *Language Tool Python*[32] hat sich als das leistungsstärkste Tool herausgestellt. Dieser Korrektor erkennt auch stilistische Fehler, wie jene, die durch die *Spaceinf*-Attacke eingeführt werden.

3.3.3. Robustheit durch Zertifizierung

Die Robustheit durch Zertifizierung wird mit dem von *Zeng et al.*[33] beschriebenen Ansatz der *Randomized Mask* implementiert. *Mask* ist hierbei das *Maskierungstoken* der *LLMs*, welches während des Trainings benutzt wird. Dementsprechend wird das *Maskierungstoken* auf das entsprechende vortrainierte Modell des Detektors angepasst. 15% der *Eingabetoken* werden zufällig ausgewählt und durch das *Maskierungstoken* ersetzt, um für jeden Text fünf Variationen zu erstellen. Die endgültige Klasse des Textes wird durch ein Mehrheitsvotum der einzelnen Ergebnisse der Variationen ermittelt.

[29] Felder: Homoglyphs, 2022.
[30] Garbe: Symspellpy, 2022.
[31] Barrus: Pyspellchecker, 2023.
[32] Myint: Checker, 2022.
[33] Zeng u. a.: Mask, 2021.

4. Experiment

In dem vorliegenden Kapitel wird der Prozess zur Erstellung eines robusten Detektors zur Erkennung von *KI*-generierten Texten in deutscher Sprache validiert und die Effektivität einzelner Verfahrensschritte analysiert. Dafür wird zunächst der Aufbau des Experimentes beschrieben und die genutzten Bewertungsmetriken erläutert. Auf Basis dieser werden die Ergebnisse der sieben Experimente anschließend dargestellt.

4.1. Experimenteller Aufbau

Basierend auf den herausragenden Ergebnissen von *Guo et al.*[1] mit einem englischen Datensatz und unter Berücksichtigung vergleichbarer Hyperparameter-Werte anderer *SOTA*-Modelle[2], dienen diese Werte als Ausgangsbasis für die in dieser Arbeit gewählten Hyperparameter. Nach einer Feinabstimmung der Hyperparameter wurde die folgende Konfiguration gewählt:

Der *Adam*-Optimierer wird mit einer Lernrate von $2e^{-5}$ eingesetzt. Die *Batch*-Größe wurde auf 32 festgelegt und das vortrainierte *LLM* wird in nur einer Epoche angepasst. Eine Erhöhung der Epochenanzahl zeigte keine verbesserten Performance-Ergebnisse. Im Gegenteil, es trat ein *Overfitting* bezüglich der Trainingsdaten auf, welches die Genauigkeit auf den Testdaten beeinträchtigte.

Die Experimente werden in einer virtuellen Umgebung auf einem lokalen Rechner durchgeführt. Dieser Rechner ist mit einem *AMD Ryzen 7 5700x* Prozessor, einer *NVIDIA GeForce RTX 4070* und 32 GB RAM ausgestattet. Für die Implementierung wurde die Programmiersprache *Python* verwendet. Die für

[1] Guo u. a.: Close, 2023, S. 11f.
[2] Solaiman u. a.: Impacts, 2019.

dieses Projekt verwendeten Bibliotheken sind in der Datei *requirements.txt* auf der beiliegenden CD aufgelistet. Eine detaillierte Dokumentation des Projektcodes findet sich im Anhang A.2.

4.2. Bewertungsmetriken

In diesem Kapitel werden die Metriken zur Bewertung der Leistung der Detektoren bei der Klassifikation zwischen menschlich verfassten und *KI*-generierten Texten beschrieben. Für diese binäre Klassifikationsaufgabe werden die folgenden Bewertungsmetriken herangezogen: *Recall*, F1-Maß und Spezifität.

Der *Recall* beziehungsweise die Sensitivität misst die Anzahl der tatsächlich *KI*-generierten Texte, die richtig klassifiziert wurden:

$$\text{Recall} = \frac{\text{Wahre Positive}}{\text{Wahre Positive + Falsche Negative}} \tag{4.1}$$

In dieser Arbeit wird die Präzision lediglich zur Berechnung des F1-Maßes benutzt. Die Präzision gibt an, wie viele der als *KI*-generiert klassifizierten Texte tatsächlich korrekt klassifiziert wurden:

$$\text{Präzision} = \frac{\text{Wahre Positive}}{\text{Wahre Positive + Falsche Positive}} \tag{4.2}$$

Das F1-Maß stellt das harmonische Mittel zwischen Präzision und *Recall* dar:

$$\text{F1-Maß} = \frac{2 \times (\text{Präzision} \times \textit{Recall})}{\text{Präzision + } \textit{Recall}} \tag{4.3}$$

Die Spezifität ist in dieser Klassifikation von besonderer Bedeutung, da sichergestellt werden sollte, dass so wenig menschliche Texte wie möglich fälschlicherweise als *KI*-generiert klassifiziert werden. Sie berechnet den Anteil der richtig klassifizierten menschlichen Texte:

$$\text{Spezifität} = \frac{\text{Wahre Negative}}{\text{Wahre Negative + Falsche Positive}} \tag{4.4}$$

Da das F1-Maß Präzision und *Recall* in einem einzigen Wert kombiniert, wird es als primäre Evaluationsmetrik genutzt. Diese Metrik berücksichtigt sowohl das Erkennen von *KI*-generierten Texten als auch das Vermeiden von Fehlklassifikationen menschlicher Texte.

4.3. Ergebnisse

Dieses Kapitel stellt die Ergebnisse der einzelnen Experimente dar. Prozentuale Zahlen werden auf die zweite Nachkommastelle gerundet. Zuerst werden die Leistungen verschiedener Modelle gegenübergestellt. Die beiden darauf folgenden Experimente untersuchen die Generalisierbarkeit des Modells hinsichtlich unterschiedlicher *Prompts* und Textarten aus diversen Bereichen. Dabei wird auch die Robustheit des Modells gegenüber *OOD*-Datenpunkten geprüft. Dies geschieht, indem die Performance des Modells auf bisher unbekannten Textarten sowie auf Texten, die durch nicht-gesehene *Prompt*-Vorlagen generiert wurden, evaluiert wird. Abschließend wird die Effektivität der in Kapitel 2.3.1 vorgestellten Attacken und die Erhöhung der Robustheit des Modells durch die in Kapitel 2.3.2 diskutierten Gegenmaßnahmen betrachtet.

4.3.1. Vortrainierter Encoder-Transformer

Die analysierten Modelle nutzen vortrainierte *LLMs* wie *BERT* und *RoBERTa*, die in der Textklassifikation als Standard gelten. Besonders im Fokus stehen dabei Sprachmodelle, die durch *Fine-Tuning* auf deutschsprachigen Texten optimiert wurden. Es wurde ausschließlich die *Base*-Variante dieser Modelle in Betracht gezogen, da sie weniger Parameter aufweisen, was zu einem reduzierten Bedarf an Rechenressourcen und kürzeren Trainingszeiten führt. Tabelle 4.1 präsentiert die durchschnittlichen Bewertungsergebnisse nach drei Durchläufen für die verschiedenen Modelle.

Ziel der Evaluation ist es, zu ermitteln, wie gut die Modelle zwischen *KI*-generierten und von Menschen verfassten Texten unterscheiden können.

Metriken → Modell ↓	F1-Maß	*Recall*	Spezifität
GBERT[3]	97,89	97,71	98,08
DBMDZ-BERT[4]	97,35	98,03	96,55
GottBERT[5]	97,38	96,84	97,80
WECHSEL-RoBERTa[6]	97,62	98,09	97,05

Tabelle 4.1.: F1-Maß (%), *Recall* (%) und Spezifität (%) des Detektors in Abhängigkeit des vortrainierten *LLMs*.

Das von *Deepset* entwickelte Modell *GBERT*[7] zeigt konstant hohe Leistungswerte in allen drei Bewertungsmetriken. Mit einem F1-Maß von 97,89% und einer Spezifität von 98,08% erzielt *GBERT* die besten Ergebnisse in diesen Kategorien. Bei der Betrachtung des *Recalls* schneiden das *WECHSEL-RoBERTa*-Modell[8] und das *DBMDZ-BERT*-Modell[9] mit 98,09% beziehungsweise 98,03% am besten ab. Da das F1-Maß als primäre Evaluationsmetrik für die Modellleistung dient, wird *GBERT* als das primäre Modell in den nachfolgenden Experimenten eingesetzt.

4.3.2. Generalisierbarkeit

In den folgenden Experimenten wird die Generalisierbarkeit des Detektors untersucht, indem die Trainingsdaten sukzessiv um neue Daten erweitert werden. Aufgrund des langen und rechenintensiven Trainings werden nicht alle Kombinationen getestet. Die Testdaten bleiben während der gesamten Untersuchung konstant, um die Performance für *OOD*-Texte konsistent zu bewerten.

[7] Chan / Schweter / Möller: Next, 2020.
[8] Minixhofer / Paischer / Rekabsaz: WECHSEL, 2021.
[9] Schmid: German, 2019.

Generalisierbarkeit des Detektors bezüglich unterschiedlicher Textarten

Als Grundlage des Trainings werden zu Beginn ausschließlich Zeitungsartikel in die Trainingsdaten aufgenommen. Dies liegt daran, dass sie den umfangreichsten Datensatz aufgrund ihrer Textlänge liefern. In den nachfolgenden Schritten wird dieser Datensatz durch das Hinzufügen weiterer Textgenres sukzessiv erweitert. Die detaillierten Ergebnisse dieses schrittweisen Prozesses sind in Tabelle 4.2 dargestellt. Die Performance des Detektors wird in dieser Tabelle durch das F1-Maß in Prozent angegeben.

Test → Train ↓	ZA	LA	KA	FA	DB	PB	FB	TW
Zeitungsartikel	96,48	93,02	51,55	88,51	89,38	85,89	76,21	59,57
+ Lexikonartikel	95,01	97,58	62,62	93,71	88,50	81,48	75,19	60,36
+ Kinderlexikonartikel	91,65	93,73	81,80	96,27	88,28	80,09	87,08	64,11
+ Fachartikel	97,89	98,66	91,41	99,08	87,31	73,83	87,13	72,46
+ Dienstleistungsbew.	95,74	98,52	95,49	**99,52**	97,31	91,36	90,16	69,71
+ Produktbewertungen	**99,59**	**99,43**	**98,41**	95,54	**97,46**	**95,54**	93,76	78,56
+ Forenbeiträge	97,25	97,93	90,12	98,85	97,07	93,93	**96,49**	93,93
+ Tweets	97,12	98,79	93,96	93,33	97,19	94,32	96,41	**98,04**

Tabelle 4.2.: F1-Maße (%) des Detektors auf den Testdaten, in Abhängigkeit der verwendeten Textarten im Training. Diese werden sukzessiv um weitere Textgenres erweitert. Die Performance wird auf den Testdaten, gruppiert nach den Textarten, untersucht.

Die Tabelle 4.2 zeigt die Leistungsvariation des Detektors in Abhängigkeit von den in den Trainingsdaten enthaltenen Textgattungen. Dabei wird die Performance auf den Testdaten gemessen, die entsprechend der Textgattungen kategorisiert sind. Die folgenden Textarten werden in der Analyse berücksichtigt: Zeitungsartikel, Lexikonartikel, Kinderlexikonartikel, Fachartikel, Dienstleistungsbewertungen, Produktbewertungen, Forenbeiträge und *Tweets*.

Bei einer anfänglichen Beschränkung der Trainingsdaten auf ausschließlich Zeitungsartikeln erreicht das Modell beachtliche F1-Maße bei der Klassifikation der meisten Textgenres. Beispielsweise sind es 96,48% bei Zeitungsartikeln und 93,02% bei Lexikonartikeln. Jedoch bleibt die Klassifikation von *Tweets* mit einem F1-Maß von 59,57% hinter den Erwartungen zurück. Bei jeder Hinzunahme einer neuen Textgattung zu den Trainingsdaten zeigt sich generell eine erhöhte Performance bei der Klassifikation von Texten der hinzugefügten Gattung. Besonders bemerkenswert ist der signifikante Anstieg des

F1-Maßes für Genres, die zuvor nur unzureichend erkannt wurden, wie zum Beispiel Kinderlexikonartikel, deren F1-Maß um etwa 19,18% steigt, sobald sie den Trainingsdaten hinzugefügt werden. Ähnlich steigt das F1-Maß für *Tweets* von 78,45% auf 93,93% an, wenn Foreneinträge in die Trainingsdaten einfließen, ein Anstieg, der sogar höher ausfällt als bei direkter Integration von *Tweets*.

Weitere Untersuchungen offenbaren einen generell positiven Trend bei der Veränderung der F1-Maße über alle Textgenres hinweg bei Hinzunahme neuer Textgenres, zumindest bis zur Einbindung der Produktbewertungen. Der Detektor, der ohne *Tweets* und Forenbeiträge trainiert wurde, zeigt durchgängig die besten F1-Maße für die meisten Textarten. Nur die Werte für Forenbeiträge und *Tweets* selbst verbessern sich durch ihre spätere Integration, im Genauen von 93,76% beziehungsweise 78,56% auf 96,41% beziehungsweise 98,04%.

Im Gegensatz dazu, gibt es auch vereinzelte Performance-Rückgänge durch die Integration neuer Textarten. Das F1-Maß für Fachartikel fällt von 99,52% auf 95,54% mit Integration der Produktbewertungen. Zudem führt die Aufnahme von Forenbeiträgen zu einem Rückgang des F1-Maßes auf Daten dieser fünf Textarten: Zeitungsartikel, Lexikonartikel, Kinderlexikonartikel, Dienstleistungsbewertung und Produktbewertungen.

Generalisierbarkeit des Detektors bezüglich unterschiedlicher Prompt-Varianten

Die Tabelle 4.3 illustriert die Performance des Detektors auf verschiedenen Segmenten der Testdaten, die nach den einzelnen *Prompts* gruppiert sind. Menschliche Daten ohne *Prompts* werden in die jeweilige Kategorie der Testdaten eingeteilt, in der sich der KI-generierte Gegenpart befindet. Dieses Vorgehen ermöglicht, mithilfe des F1-Maßes sowohl die Trefferquote von *KI*-generierten Texten als auch die Fehlerquote bei Texten, die von Menschen verfasst wurden, zu bewerten. Zur Evaluierung der Performance auf verschiedenen *Prompt*-Segmenten der Testdaten werden die Trainingsdaten schrittweise um weitere *Prompt*-Variationen erweitert.

In Tabelle 4.3 stellen die Spalten Segmente von Testdaten dar. Diese Daten wurden basierend auf der jeweiligen *Prompt*-Vorlage erstellt und sind mit dem entsprechenden menschlichen Text verknüpft.

Test → Train ↓	A	B	C	D	E	F	G
Prompt A	97,50	97,75	97,68	98,16	97,57	97,64	99,01
+ Prompt B	96,57	97,13	96,86	97,02	96,53	97,16	98,75
+ Prompt C	99,57	**99,25**	**99,61**	**99,47**	**99,82**	**99,89**	**99,86**
+ Prompt D	99,11	98,53	99,29	99,32	99,43	99,32	99,68
+ Prompt E	98,78	98,46	98,75	99,11	99,25	99,14	99,68
+ Prompt F	**98,82**	98,47	98,65	99,17	99,05	99,21	99,57
+ Prompt G	98,76	98,44	98,52	99,09	99,23	99,12	99,66

Tabelle 4.3.: F1-Maße (%) des Detektors auf den Testdaten, in Abhängigkeit der verwendeten *Prompts* im Training. Die Trainingsdaten werden schrittweise durch verschiedene *Prompt*-Variationen ergänzt. Die Performance des Detektors wird in verschiedenen Segmenten der Testdaten, die nach den einzelnen *Prompt*-Varianten gruppiert sind, bewertet.

Wird das Modell ausschließlich auf Daten des *Prompts A* trainiert, ergeben sich F1-Maße von 97,50% für Segment A, 97,75% für Segment B, 97,68% für Segment C und bis zu 99,01% für Segment G. Das Modell zeigt demnach durchweg gute Leistungen auf allen Segmenten. Bei Einbindung der Daten des zweiten *Prompts* in die Trainingsdaten lässt sich ein leichter Rückgang des F1-Maßes über alle Segmente hinweg beobachten. Die größte Performancesteigerung des Modells wird erzielt, wenn die Daten des *Prompts C* integriert werden. An diesem Punkt ist ein deutlicher Anstieg des F1-Maßes zu erkennen; die Werte liegen bei 99,25% für Segment B und erreichen einen herausragenden Wert von 99,89% für Segment E. Bei Integration der Daten von *Prompt D* bis *Prompt G* in die Trainingsdaten bleiben die F1-Maße weitgehend stabil zwischen 98% und 99%. Besonders hervorzuheben ist der Spitzenwert von 99,86% für Segment G bei dem Training mit Daten bis *Prompt C*.

4.3.3. Attacken

In diesem Unterkapitel werden die im theoretischen Kapitel beschriebenen Attacken auf *KI*-generierte Texte angewendet, um die Robustheit des Detektors zu ermitteln. Da sich dieses Experiment speziell auf die Umgehung der Klassifikation von *KI*-generierten Texten konzentriert, wird lediglich der *Recall*

auf den veränderten Testdatensätzen betrachtet. Für jede Attacke wird ein separater Testdatensatz erstellt, der ausschließlich aus den ursprünglichen *KI*-generierten Textdaten besteht, die durch die jeweilige Attacke manipuliert werden. Dabei bleibt die Architektur des zugrunde liegenden Detektors unverändert.

Die verwendeten Attacken lassen sich in zwei Kategorien unterteilen: Syntaktische und semantische Manipulationen. Der *Recall* für die einzelnen Attacken ist in Tabelle 4.4 prozentual angegeben. Dieser Wert gibt die Trefferquote der KI-generierten Texte nach Anwendung der jeweiligen Attacke an. Zudem stellt die Spalte Δ *Recall* die prozentuale Veränderung im Vergleich zum *Recall* auf dem Testdatensatz ohne Manipulationen dar.

Metriken → Attacken ↓	*Recall*	Δ *Recall*
Basisfall		
Ohne Attacke	97,71	
Syntaktische Manipulationen		
Einfügen von Homoglyphen	2,58	-95,13
Einfügen von Tippfehler	79,33	-18,38
SpaceInf	85,58	-12,13
Einfügen von Rechtschreibfehler	94,81	-2,9
Semantische Manipulationen		
Austausch durch Synonyme	89,86	-8,03
Paraphrisierung	88,03	-9,68

Tabelle 4.4.: Auswirkungen einzelner Attacken auf den *Recall* (%) des Detektors. Die Δ *Recall* (%) Spalte zeigt die prozentuale Veränderung gegenüber dem *Recall* (%) auf den Testdatensatz ohne Manipulationen.

Die Ergebnisse zeigen, dass sich syntaktische Manipulationen stärker auf den *Recall* des Detektors auswirken als semantische Manipulationen. Insbesondere die Einführung von Homoglyphen führt zu einem drastischen Rückgang des *Recalls* um 95,13%. Das Einfügen von Tippfehlern und die Anwendung der *SpaceInf*-Methode, in welcher Leerzeichen vor zwei zufälligen Kommata im Text gesetzt werden, führen zu einer Verringerung des *Recalls* um 18,38% beziehungsweise 12,13%. Die Präsenz von Rechtschreibfehlern reduziert die Erkennungsrate um 2,9%. Semantische Veränderungen, wie das Ersetzen von Wörtern durch Synonyme und das Paraphrasieren von Texten, führen zu einer Verringerung des *Recalls* um 8,03% beziehungsweise 9,68%.

4.3.4. Gegenmaßnahmen

Die Bewertung der Gegenmaßnahmen erfolgt auf Basis derselben Datensätze, die bereits im vorangegangenen Experiment zur Untersuchung der Attacken verwendet wurden. Der Unterschied besteht darin, dass Verteidigungsstrategien zur robusten Detektion implementiert werden.

Adversarial Training

Im *Adversarial Training* wird das Modell nicht nur mit dem herkömmlichen Trainingsdatensatz, sondern auch mit *Adversarial Examples* trainiert. Diese werden durch Anwendung der im vorherigen Unterabschnitt beschriebenen Attacken auf Teilen der Trainingsdaten generiert. Im Testdatensatz befinden sich unbekannte *Adversarial Examples* jeder Attacke. Da diese Methode den Trainingsprozess modifiziert, wird ein neues Modell mit den Daten, einschließlich der *Adversarial Examples*, trainiert.

Die Performance des Detektors in der Erkennung von *KI*-generierten Texten ist in Tabelle 4.5 dargestellt. Die Messung der Performance erfolgt mit dem F1-Maß und dem *Recall*. Das F1-Maß wird verwendet, um sicherzustellen, dass die Fehlklassifikationsrate bei menschlichen Texten entsprechend gering ist. Der primäre Fokus liegt jedoch auf dem *Recall*, da die Attacken darauf abzielen, diesen Wert zu reduzieren. Die Änderungen in *Recall* und F1-Maß beziehen sich auf die Basiswerte nach Anwendung der Attacken auf den Testdatensatz, die in Tabelle 4.4 dargestellt sind.

Das *Adversarial Training* steigert den *Recall* und damit die Trefferquote von manipulierten, *KI*-generierten Texten. Dieser Effekt ist bei jeder Attacke feststellbar. Die Erkennungsrate liegt für Texte, die durch das Hinzufügen von Homoglyphen, Tippfehlern, zusätzlichen Leerzeichen, Rechtschreibfehlern oder das Ersetzen von Wörtern durch Synonyme manipuliert wurden, bei über 99%. Der *Adversarial Trainings*-Ansatz kompensiert nicht nur die Effekte der Attacken, er führt sogar zu leicht verbesserten Erkennungsraten für die manipulierten, *KI*-generierten Texte. Dies resultiert in einer verbesserten Leistung des Detektors, was durch den Anstieg des F1-Maßes bei diesen Attacken verdeutlicht wird.

Obwohl das *Adversarial Training* die Erkennung von paraphrasierten *KI*-generierten Texten um 6,62% steigert, erreichen der *Recall* und das F1-Maß nicht die ursprünglichen Werte von vor der Attacke.

Metriken → Attacken ↓	*Recall*	Δ *Recall*	F1-Maß	Δ F1-Maß
Homoglyphen	99,99	+97,41	98,36	+93,42
Tippfehler	99,98	+20,65	98,35	+10,81
SpaceInf	99,61	+14,03	98,17	+7,24
Rechtschreibfehler	99,6	+4,79	98,16	+1,82
Synonyme	99,28	+9,42	98	+4,29
Paraphrisieren	94,65	+6,62	95,61	+1,41

Tabelle 4.5.: Robustheit des Detektors bei Anwendung des *Adversarial Trainings*. Die Robustheit wird durch das F1-Maß (%) sowie den *Recall* (%) gemessen. Die Spalten Δ *Recall* (%) und Δ *F1-Maß* (%) zeigen die prozentuale Veränderung gegenüber den beiden Werten auf den manipulierten Testdatensätzen ohne Gegenmaßnahme.

Korrektur von Perturbationen

Die Korrektur von Perturbationen erfolgt durch den Autokorrektor *Language Tool Python* und ein selbst entworfenes Programm zur Korrektur von Homoglyphen. Da diese Korrekturen primär syntaktische Manipulationen betrachten, wird lediglich die Effektivität der Korrektur von Perturbationen für diese spezifischen Attacken evaluiert. Das Modell selbst wird nicht modifiziert, da die Korrekturen lediglich während der Inferenz integriert werden.

Die Tabelle 4.6 folgt dem Aufbau der im vorherigen Unterabschnitt beschriebenen Tabelle 4.5.

Metriken → Attacken ↓	*Recall*	Δ *Recall*	F1-Maß	Δ F1-Maß
Homoglyphen	97,71	+95,13	97,89	+92,95
Tippfehler	90,31	+10,98	93,7	+6,16
SpaceInf	97,21	+11,63	97,37	+6,44
Rechtschreibfehler	97,46	+2,65	97,5	+1,16

Tabelle 4.6.: Robustheit des Detektors bei Anwendung der Korrektur von Perturbationen. Die Robustheit wird durch das F1-Maß (%) sowie den *Recall* (%) gemessen. Die Spalten Δ *Recall* (%) und Δ *F1-Maß* (%) zeigen die prozentuale Veränderung gegenüber den beiden Werten auf den manipulierten Testdatensätzen ohne Gegenmaßnahme.

Dieser Verteidigungsansatz zeigt bei jeder der betrachteten Attacken beeindruckende Ergebnisse. Die Auswirkungen von Homoglyphen, durch die *SpaceInf*-Attacke eingefügten Leerzeichen und Rechtschreibfehler auf den *Recall* des Detektors werden nahezu vollständig durch die Korrektur von Perturbationen eliminiert. Der Recall steigt bei den durch Tippfehler und *SpaceInf* manipulierten Texten um 10,98% beziehungsweise 11,63%. Bei durch Homoglyphen beeinflussten Texten wird sogar ein Anstieg von 95,13% verzeichnet. Infolgedessen bleibt das F1-Maß des Detektors für diese Attacken konstant über 97%. Durch Tippfehler beeinflusste *KI*-generierte Texte werden in 10,98% der Fälle durch die Korrektur von Perturbationen korrekt identifiziert. Dennoch erreichen der Recall und das F1-Maß auf diesen Texten nicht ihre ursprünglichen Werte.

Robustheit durch Zertifizierung

In dieser Arbeit wird das Konzept der *Randomized Masks* als Zertifizierung genutzt. Da dieser Ansatz eine mehrfache Klassifikation der Texte erfordert, kann der Detektor nicht mehr lokal in einer angemessenen Zeit trainiert werden. Deswegen wird der *Randomized Mask*-Ansatz lediglich während der Inferenz angewendet. Tabelle 4.7 zeigt den *Recall,* das F1-Maß und die Veränderungen dieser Metriken nach Implementierung der *Randomized Masks*.

Metriken → Attacken ↓	*Recall*	Δ *Recall*	F1-Maß	Δ F1-Maß
Homoglyphen	1,5	-1,03	3,05	-1,89
Tippfehler	49,86	-29,47	66,38	-21,16
SpaceInf	96,86	+11,28	98,23	+6,94
Rechtschreibfehler	77,46	-17,35	87,11	-9,28
Synonyme	67,24	-22,62	80,23	-13,48
Paraphrisieren	93,52	+5,49	96,48	+2,28

Tabelle 4.7.: Robustheit des Detektors bei Anwendung der *Randomized Masks*. Die Robustheit wird durch das F1-Maß (%) sowie den *Recall* (%) gemessen. Die Spalten Δ *Recall* (%) und Δ *F1-Maß* (%) zeigen die prozentuale Veränderung gegenüber den beiden Werten auf den manipulierten Testdatensätzen ohne Gegenmaßnahme.

Die Implementierung dieser Verteidigungsmaßnahme zeigt unterschiedliche Effekte auf verschiedene Attacken. Die Erkennung von durch Tippfehler manipulierten Texten verringerte sich signifikant um 29,47%. Gleichzeitig verschlechterte sich die Performance des Modells, gemessen am F1-Maß, um 21,16% auf 66,38%. Ähnliche Einbußen zeigen sich bei den durch Rechtschreibfehler und Synonymen manipulierten Texten. Interessanterweise ging auch die Erkennungsrate von mit Homoglyphen versehenen *KI*-generierten Texten von 2,53% auf 1,5% zurück.

Dennoch weist die Verteidigungsstrategie gegen die *Space-Inf*- und Paraphrasierungs-Attacken eine erhöhte Robustheit auf. Die Erkennungsrate für diese steigt um 11,28% beziehungsweise 5,49%. Dies führt zu herausragenden F1-Maßen von 96,48% für umformulierte Texte und sogar 98,23% für durch die *Space-Inf*-Attacke manipulierten Texte.

5. Diskussion

In diesem Kapitel wird über die Bedeutung der experimentellen Ergebnisse diskutiert. Verschiedene Modelle werden hinsichtlich ihrer Stärken und Schwächen analysiert. Zudem wird die Generalisierbarkeit des Detektors über die Trainingsdaten hinweg bewertet. Es wird auch die Effektivität von Angriffen auf den entwickelten Detektor sowie die Wirksamkeit von Gegenmaßnahmen betrachtet. Die in dieser Arbeit vorgestellten Ergebnisse liefern neue Einsichten in den Forschungsbereich der Erkennung von *KI*-generierten Texten, insbesondere im Hinblick auf die Verbesserung der Robustheit der Detektoren.

5.1. Modellauswahl

Wie das Experiment aus Abschnitt 4.3.1 zeigt, sind alle untersuchten Modelle in der Lage, *KI*-generierte Texte in deutscher Sprache mit hoher Präzision zu identifizieren. Das *GBERT*-Modell sticht hierbei besonders hervor. Seine hohe Spezifität deutet darauf hin, dass es menschliche Texte nur selten fälschlicherweise als *KI*-generiert einstuft. Während sowohl *DBMDZ-BERT* als auch *WECHSEL-RoBERTa* *KI*-generierte Texte präzise erkennen, tendieren sie dazu, menschliche Texte häufiger fehlzuklassifizieren. Dies wird besonders beim *WECHSEL-RoBERTa* deutlich, dessen *Recall* den aller anderen im Experiment untersuchten *BERT*-basierten Modelle übertrifft. Hierbei könnten architektonische Unterschiede der Grund für den hohen *Recall* von *WECHSEL-RoBERTa* sein.

Insgesamt deuten die Ergebnisse darauf hin, dass es trotz beeindruckender Leistung aller Modelle, Unterschiede im *Recall* und der Spezifität vorhanden sind. Dies zeigt, dass jedes Modell unterschiedliche Prioritäten hinsichtlich

des Erkennens von *KI*-generierten Texten und des Vermeidens von Fehlklassifikationen menschlicher Texte setzt. Das *GBERT*-Modell wird aufgrund seiner ausgewogenen Leistung, die durch das hohe F1-Maß belegt wird, für Standardimplementierungen empfohlen. Die in der untersuchten Literatur hervorgehobene Bedeutung einer geringen Fehlklassifikationsrate bestätigt die Stärke des *GBERT*-Modells.[1]. Sollte in einem bestimmten Anwendungsfall die Fehlklassifikation menschlicher Texte weniger kritisch betrachtet werden, so bietet das *WECHSEL-RoBERTa*-Modell eine geeignete Alternative.

5.2. Generalisierbarkeit

Generalisierbarkeit bezüglich unterschiedlicher Textgattungen: Der Performancegewinn durch die Hinzunahme von Textarten deutet darauf hin, dass die Diversität und das Volumen der Daten die Erkennung von *KI*-generierten Texten optimieren. Insbesondere die Integration von Daten ähnlicher Textarten, wie beispielsweise Produkt- und Dienstleistungsbewertungen, oder besser noch derselben Textgattung, führen zu signifikanten Performanceverbesserungen. Dies war zu erwarten, da jede Textgattung spezifische Merkmale aufweist. Dementsprechend müssen die für eine solche Textart charakteristischen Muster, sei es von *KI*-generierten oder von Menschen erstellten Texten, erlernt werden. Wenn die zu klassifizierende Textart festgelegt ist, kann der Detektor effizienter auf diese spezielle Textgattung trainiert werden.

Besonders im Hinblick darauf, dass das Hinzufügen unterschiedlicher Textarten auch zu einer verringerten Klassifizierungsleistung führen kann, insbesondere wenn die Textarten stark voneinander abweichen. So beeinträchtigte beispielsweise die Integration von umgangssprachlichen Forenbeiträgen in den Trainingsdatensatz die Erkennungsleistung bei wissenschaftlich und neutral formulierten Lexikon- und Zeitungsartikeln.

Aus diesen Ergebnissen ergibt sich die Empfehlung, den Detektor gezielt auf Textarten mit ähnlichem Aufbau und Sprachgebrauch zu trainieren, um optimale Ergebnisse für den jeweiligen Anwendungsbereich zu erzielen.

[1] Solaiman u. a.: Impacts, 2019.

Generalisierbarkeit des Detektors bezüglich unterschiedlicher *Prompts*: Der Detektor besitzt beeindruckende Fähigkeiten bei der Erkennung von *KI*-generierten Texten, selbst wenn er nur auf Daten einer spezifischen *Prompt*-Vorlage trainiert wird. Daraus lässt sich schließen, dass die Modifikation des *Prompts* die Erkennung nicht negativ beeinflusst. Diese Beobachtung deckt sich mit den Ergebnissen von Cai und Cu[2] und widerspricht gleichzeitig den Untersuchungen von *Liang et al.*[3] In dieser Arbeit wurden verschiedene Ansätze getestet, die Erkennung mithilfe von *Prompts* zu umgehen, welche sich jedoch durchweg als unwirksam herausstellten. Ein Grund für die robuste Erkennung trotz diverser *Prompt*-Variationen könnte das Satz-für-Satz *Tokenisieren* im Training sein.

Ein weiterer signifikanter Befund ist der deutliche Anstieg des F1-Wertes mit dem Hinzufügen von Daten des *Prompts C*. Dieser *Prompt* gibt eine durchschnittliche Textlänge für die Modellausgaben vor, die der durchschnittlichen Länge von menschlich verfassten Texten dieser Textart entspricht. Die Hinzunahme dieser Trainingsdaten könnte zur Robustheit des Modells in Bezug auf variable Textlängen beigetragen haben.

Zusammenfassend zeichnet der Detektor sich durch eine ausgezeichnete Generalisierbarkeit bei variierenden *Prompts* aus. Die Resultate könnten ein Indiz dafür sein, dass nicht alle Trainingsdaten für die Entwicklung eines effektiven Detektors benötigt werden.

Limitationen: Eine Limitation dieser Experimente ist, dass lediglich eine festgelegte Reihenfolge bei der Hinzunahme von Textgattungen und *Prompts* in Betracht gezogen wurde. Daher besteht die Möglichkeit, dass Nebeneffekte die Ergebnisse beeinflusst haben.

Darüber hinaus sollten bei dem Experiment zur Untersuchung der Generalisierbarkeit des Detektors bezüglich der verschiedenen *Prompts* berücksichtigt werden, dass keine extremen Stiländerungen durch den *Prompt* durchgeführt wurden. Ein Beispiel dafür wäre, das generierende *LLM* aufzufordern einen Text im Stil von *Shakespeare* zu schreiben. Der Grund hierfür ist, dass solche Modifikationen in realen Anwendungsszenarien oftmals keinen Mehrwert bieten.

[2] Cai / Cui: Evade, 2023, S. 5.
[3] Liang u. a.: Detectors, 2023, S. 2.

5.3. Effektivität der Attacken

Die untersuchten Attacken können die Erkennung von *KI*-generierten Texten signifikant reduzieren. Besonders Homoglyphen erzielen in dieser Hinsicht beeindruckende Resultate. Durch die Integration von Homoglyphen konnte die Erkennung von *KI*-generierten Texten in 95% der Fälle verhindert werden. Das bemerkenswerte an dieser Methode ist, dass sie für den Menschen unsichtbar bleibt, da Homoglyphen des kyrillischen und griechischen Alphabets von uns als identisch zu den lateinischen Buchstaben wahrgenommen werden. Für maschinelle Systeme werden sie jedoch unterschiedlich interpretiert. Dies bedeutet, dass die Detektoren einen Text mit für sie unbekannten *Tokens* vorfinden und daher keine typischen Muster von *KI*-generierten Texten identifizieren können. Die hohe Erfolgsrate dieser Methode stimmt mit den Ergebnissen von *Wolff* und *Wolff*[4] überein, die eine Reduktion des *Recalls* des Detektors von 97,44% auf 13,57% durch die Verwendung kyrillischer Homoglyphen feststellten.

Tipp- und Rechtschreibfehler beeinflussen die *Tokens* nicht in dem Maße wie Homoglyphen. Ein zu häufiges Anwenden von Tipp- und Rechtschreibfehlern kann auffällig sein und die Qualität des Textes erheblich beeinträchtigen. Daher wurden sie in den Experimenten nicht in der gleichen Frequenz wie Homoglyphen verwendet. Das bewusste Einfügen von Rechtschreibfehlern zeigte in den durchgeführten Experimenten keine hohe Erfolgsrate. Möglicherweise liegt dies daran, dass nicht ausreichend Rechtschreibfehler eingefügt wurden, weil die Liste der häufigen Rechtschreibfehler nicht genügend Fehlervarianten beinhaltete.

Das Paraphrasieren konnte zwar die Erkennung einiger *KI*-generierter Texte verhindern, erreichte in den durchgeführten Experimenten dieser Arbeit jedoch eine geringere Erfolgsrate im Vergleich zu anderen wissenschaftlichen Untersuchungen. *Krishna et al.*[5] konnten die Genauigkeit des Detektors um 50% reduzieren. Diese Abweichung könnte auf unterschiedliche Implementierungsansätze des Paraphrasierens zurückzuführen sein. Während *Krishna et al.* ein spezialisiertes Modell namens *Dipper* für ihre Attacke verwendeten, wurde in dieser Arbeit ein Ansatz basierend auf dem Zurückübersetzen gewählt. Obwohl *Dipper* aufgrund seines spezifischen Designs Vorteile bietet,

[4] Wolff / Wolff: Attacking, 2020, S. 3.
[5] Krishna u. a.: Paraphrasing, 2023.

repräsentiert das Zurückübersetzen realistischere Angriffsszenarien, da im deutschsprachigen Raum ein leichterer Zugang zu Übersetzungstools besteht als zu spezialisierten Modellen für das Paraphrasieren.

Abschließend lässt sich sagen, dass bereits einfache Methoden die Effektivität des Detektors signifikant verringern können. In dieser Arbeit wurden jedoch keine Kombinationen der unterschiedlichen Angriffsmethoden betrachtet. Zudem könnten die Veränderungen durch die Attacken noch intensiver ausfallen, wobei hier zu berücksichtigen ist, dass die Integrität des Textes nicht beeinträchtigt werden sollte. Zukünftige Experimente könnten eine zusätzliche Metrik einführen, die die Qualität des Textes misst, damit die maximale Anzahl an Modifikationen in dem Text durchgeführt werden können, ohne seine Integrität zu beeinträchtigen.

5.4. Wirksamkeit der Gegenmaßnahmen

***Adversarial Training*:** Das *Adversarial Training* stellt eine effektive Gegenmaßnahme gegen sowohl syntaktische als auch semantische Manipulationen dar. Dank dieser Maßnahme ist der Detektor nicht nur in der Lage, manipulierte Texte zu erkennen, sondern erzielt dabei sogar bessere Ergebnisse als bei der Erkennung von nicht manipulierten Texten. Vermutlich liegt dies daran, dass der Detektor Homoglyphen und spezielle Tippfehler als Merkmale für die Erkennung dieser Texte nutzt, was den hohen *Recall* von fast 100% in beiden Fällen erklärt. Zudem scheint die *Keyboard Aug*-Klasse der *NLPAug*-Bibliothek häufig die gleichen Tippfehler zu simulieren. Der entwickelte Detektor zeigt jedoch eine geringere Robustheit gegenüber dem Paraphrasieren von Texten, da hier lediglich ein leichter Anstieg des *Recalls* nach dem Einsatz des *Adversarial Trainings* zu erkennen ist.

Ein wesentlicher Nachteil dieser Gegenmaßnahme ist, dass alle *Adversarial Attacks* bekannt sein müssen. Die in diesem Experiment erzielten Ergebnisse sind möglicherweise so hoch, weil exakt die gleichen Modifikationen für das Training verwendet wurden. Daher könnte die Effizienz dieser Gegenmaßnahme in realen Szenarien geringer ausfallen.

Korrektur von Perturbationen: Die Korrektur von Perturbationen ist effektiv gegen alle syntaktischen Manipulationen. Diese Maßnahme lässt sich ohne großen Aufwand implementieren, da sie direkt in den Inferenzprozess integriert werden kann, ohne die Architektur des Modells zu ändern. Zudem ist sie wahrscheinlich robuster gegenüber allgemeineren Angriffen im Vergleich zum *Adversarial Training*. Der Autokorrektor *Language Tool Python* kann eine Vielzahl weiterer Fehler korrigieren und hat sich als effektiv für die Abwehr von Angriffen erwiesen.

Robustheit durch Zertifizierung: Der *Randomized Masks*-Ansatz verstärkt einige Angriffe sogar. Dies könnte daran liegen, dass während der Inferenz möglicherweise die falschen *Tokens* maskiert werden. Dabei sollte berücksichtigt werden, dass dieses Problem auch mit der in dieser Arbeit genutzten Implementierung der Strategie zusammenhängen könnte, da sie nur während der Inferenz angewendet wurde. Trotz dieser Schwäche zeigt der Ansatz bei der Paraphrasierung von Texten eine Steigerung der Robustheit, wie durch das hohe F1-Maß belegt wird. Angesichts der Schwächen des *Adversarial Trainings* im Umgang mit paraphrasierten Texten, könnte dieser Ansatz eine sinnvolle Ergänzung darstellen. Es sollte jedoch durch einen Autokorrektor, wie in der Korrektur von Perturbationen, sichergestellt werden, dass keine anderen Manipulationen vorliegen, da diese möglicherweise verstärkt werden könnten.

6. Fazit

In diesem Kapitel werden die Erkenntnisse dieser Thesis zusammengefasst. Durch eine Reflexion werden diese Ergebnisse im Kontext ihrer Anwendbarkeit in realen Szenarien bewertet. Abschließend werden Überlegungen zu zukünftigen Herausforderungen in diesem Forschungsbereich dargestellt.

6.1. Zusammenfassung

Die vorliegende Arbeit bietet einen detaillierten Überblick über die Landschaft der aktuellen *LLMs* und der entsprechenden Methoden zur Erkennung von Texten, die von diesen *LLMs* generiert werden. In diesem Kontext hat sich die Robustheit der Detektoren als eine Kernherausforderung der *SOTA*-Detektoren herausgestellt. Außerdem bleibt die Erkennung von nicht-englischsprachigen *KI*-generierten Texten fast unerforscht, während die Fähigkeit der *LLMs*, menschenähnliche Texte in diversen Sprachen zu generieren, stetig voranschreitet. Deshalb wurde in dieser Thesis ein robuster Detektor zur Erkennung von *KI-generierten* Texten in deutscher Sprache entwickelt. In diesem Prozess hat sich die Erstellung eines qualitativ hochwertigen Datensatzes für den deutschsprachigen Raum als Schlüsselkomponente herausgestellt. Auf Grundlage dieser Daten wurde ein Detektor entwickelt und evaluiert, welcher in der Lage ist, effizient deutsche *KI*-generierte Texte zu erkennen. Darüber hinaus wurde nicht nur die Generalisierbarkeit dieses Detektors validiert, sondern auch Ansätze zur Steigerung seiner Robustheit präsentiert. Diese Ansätze ermöglichen es, sowohl syntaktische als auch semantische Manipulationsversuche, die darauf abzielen, die Klassifikation zu umgehen, erfolgreich abzuwehren.

6.2. Reflexion

Im Rückblick auf den Forschungsprozess und die daraus resultierenden Erkenntnisse erscheint es mir wichtig, einige persönliche Überlegungen und Interpretationen zu teilen. Obwohl in dieser Arbeit gezeigt wurde, dass die Erkennung von *KI*-generierten Texten eine hohe Genauigkeit erreichen kann, möchte ich betonen, dass den Detektoren nicht uneingeschränkt vertraut werden sollte. In der Regel handelt es sich bei *SOTA*-Detektoren um *Black-Box*-Systeme, deren Entscheidungen weder für den Nutzer noch für den Entwickler transparent sind. Es ist beispielsweise möglich, dass häufig verwendete Formulierungen in wissenschaftlichen Arbeiten von *LLMs* übernommen werden. Dies könnte den Detektor irrtümlich zu dem Schluss führen, dass diese Standardformulierungen auf einen *KI*-generierten Text schließen lassen. Bis dato existiert keine klare Definition, ab welchem Punkt ein Text als *KI*-generiert gilt. Ist beispielsweise ein Text, bei dem *ChatGPT* lediglich einige Formulierungen optimiert hat, wobei der Inhalt vollständig vom Autor stammt, als *KI*-generiert zu definieren?

Dennoch betrachte ich die Erkennung von *KI*-generierten Texten als einen wichtigen Forschungsbereich. Dieser kann dazu beitragen, die Risiken, die durch *LLMs* entstehen, zu minimieren. Soziale Medien stehen bereits jetzt vor großen Herausforderungen durch die Verbreitung von Falschinformationen und die Präsenz zahlreicher Bots. *LLMs* könnten diese Probleme weiter verschärfen. Die Erkennung von *KI*-generierten Texten kann in diesem Kontext eine effiziente Lösung bieten. Auch in anderen Sektoren, wie dem Forschungs- und Bildungswesen, könnten solche Detektoren als zusätzliche Metrik dienen und Hinweise darauf geben, ob eine Arbeit einer intensiveren Überprüfung bedarf.

6.3. Ausblick

In der vorhergehenden Diskussion wurden bereits einige Limitationen der in dieser Studie benutzten Angriffe und Gegenmaßnahmen aufgezeigt. Weiterführende Studien könnten auf Grundlage dieser effizientere Attacken und umfangreichere Gegenmaßnahmen für die robuste Detektion entwickeln.

Bislang hat sich die Erkennung von *KI-generierten* Texten primär auf den monolingualen Bereich konzentriert. Es wäre daher von Interesse, die Effektivität von multilingualen Detektoren zu evaluieren. Ein solcher multilingualer Ansatz könnte die Detektion robuster gestalten, indem der Detektor anderssprachige Fachbegriffe in die Textklassifikation einbezieht. Zudem könnte das Modell durch den Umgang mit multilingualen Texten Konzepte über verschiedene Sprachen hinweg erfassen. Ähnliche Beobachtungen wurden von *Pires et al.*[1] in Bezug auf *Multilingual BERT* in anderen Bereichen des *NLPs* gemacht.

Ein weiterer Aspekt, der in dieser Arbeit nicht berücksichtigt wurde, ist die Fähigkeit, Texte zu erkennen, die von unterschiedlichen *LLMs* produziert werden. Angesichts der kontinuierlichen Verbesserungen von *Open-Source*-Modellen ist es entscheidend, das Generalisierungsvermögen von Detektoren gegenüber neuen *LLMs* zu verstehen. *Li et al.*[2] haben in diesem Bereich bereits vorläufige Erkenntnisse präsentiert. Zusätzlich könnte auch die Untersuchung von diversen Parametermodifikationen der generierenden *LLMs* von Interesse sein.

Im Hinblick auf die Regulierung von *LLM*-erzeugten Texten ist es zudem von Bedeutung, Klassifikationsentscheidungen von Detektoren nachvollziehbar zu gestalten. Aktuelle *SOTA*-Detektoren agieren als *Black-Box*-Systeme, weshalb ein verstärkter Fokus auf der Verbesserung ihrer Interpretierbarkeit liegen sollte. *Mitrovic et al.*[3] haben mit dem *Shapely Additive Explanations Tool* bereits erste Fortschritte in dieser Hinsicht erzielt.

[1] Pires / Schlinger / Garrette, 2019, S. 3.
[2] Li u. a.: Origin, 2023.
[3] Mitrović / Andreoletti / Ayoub: Chatgpt, 2023.

A. Anhang

A.1. Homoglyphen

Original	Homoglyph
a (U+0061)	a (U+0430)
c (U+0063)	c (U+0441)
e (U+0065)	e (U+0435)
j (U+006A)	j (U+0458)
o (U+006F)	o (U+043E)
s (U+0073)	s (U+0455)
A (U+0041)	A (U+0391)
A (U+0041)	A (U+0410)
B (U+0042)	B (U+0392)
B (U+0042)	B (U+0412)
C (U+0043)	C (U+0421)
E (U+0045)	E (U+0395)
E (U+0045)	E (U+0415)
H (U+0048)	H (U+041D)
J (U+004A)	J (U+0408)
K (U+004B)	K (U+041A)
M (U+004D)	M (U+041C)
N (U+004E)	N (U+039D)
O (U+004F)	O (U+039F)
O (U+004F)	O (U+041E)
P (U+0050)	P (U+0420)
S (U+0053)	S (U+0405)
T (U+0054)	T (U+03A4)
T (U+0054)	T (U+0422)
X (U+0058)	X (U+0425)
Z (U+005A)	Z (U+0396)

Abbildung A.1.: Homoglyphen und deren Unicode-Zeichen.

A.2. Dokumentation der Programmierleistung

In der vorliegenden Dokumentation wird der auf der beiliegenden CD bereitgestellte Code erläutert, sodass der Leser den Detektor selbst ausprobieren und die zugehörigen Programmierschritte nachvollziehen kann.

Ordnerstruktur

- **attacks/**: Enthält die Skripte, die für diverse *Adversarial Attacks* verwendet werden.
  - `common_errors.py`: Einfügen von Rechtschreib- und Tippfehlern.
  - `common_spellingerrors_de.json`: Ein Wörterbuch, das korrekte deutsche Schreibweisen und deren häufige Fehler auflistet.
  - `homoglyphs.py`: Einfügen von Homoglyphen in Texten.
  - `paraphrase.py`: Paraphrasieren von Texten.
  - `space_infiltration.py`: Einfügen von Leerzeichen vor ausgewählten Kommata in Texten.
  - `stopword_de.txt`: Liste, die häufig vorkommende deutsche Stoppwörter beinhaltet.
  - `synonym_replacement.py`: Automatisiertes Ersetzen von Wörtern durch ihre Synonyme.
- **data/**: Enthält Daten.
  - `adversarial_data/`: Enthält durch verschiedene Angriffsmethoden manipulierte Datensätze.
  - `chatGPT_data/`: Enthält *KI*-generierte Textdaten unterschiedlicher Textgenres.

  - `human_data/`: Enthält von Menschen verfasste Textdaten nach der Datenbereinigung.

  - `final_data/`: Enthält einen kombinierten Datensatz aus menschlich erstellten und *KI*-generierten Texten.

- **data_processing/**: Enthält alle Skripte und Tools, die für die Datenaufbereitung notwendig sind.

  - `data_crawler.py`: Extraktion von Fachartikeln und Daten aus *Reddit*.

  - `data_cleaning.py`: Bereinigung der jeweiligen Datensätze.

  - `generate/`: Generierung künstlicher Daten.

    * `Bard/`: Interaktion mit *Bard*.

    * `Claude2/`: Interaktion mit *Claude 2*.

    * `ChatGPT/`: Interaktion mit *ChatGPT*, inklusive der erforderlichen Prompt-Vorlagen.

- **defenses/**: Enthält die Verteidigungsmechanismen gegen die Angriffe.

  - `error_corrector/.py`: Autokorrektor.

  - `homoglyphs/`: Korrigieren von Homoglyphen.

  - Notiz: *Adversarial Training* und Robustheit durch Zertifizierung sind im Trainings- beziehungsweise Inferenzprozess des Detektors integriert.

- **detector/**: Enthält Skripte, die für die Arbeit mit dem Detektor benötigt werden.

  - `saved_models/`: Gespeicherte Detektoren (aufgrund von begrenztem Speicherplatz sind nicht alle Modelle vorhanden).

  - `eval.py`: Evaluierung der Performance der Detektoren.

- `infer.py`: Anwendung der Detektoren auf benutzerdefinierte Texte.
- `train.py`: Training des Detektors

Anleitung zur Einrichtung

Die in dieser Arbeit verwendete Programmiersprache ist *Python*. Um den gesamten Code ausführen zu können, ist mindestens *Python* Version 3.10 erforderlich. Darüber hinaus wird *CUDA 11.8* benötigt, falls der Detektor auf einer *NVIDIA* Grafikkarte trainiert werden soll.

Installieren Sie alle in der `requirements.txt`-Datei aufgeführten Pakete.

`pip install -r requirements.txt`

```
1 pip install -r requirements.txt
```

Benutzung

Mit Hilfe der *infer.py* Datei innerhalb des Ordners *detector* kann ein *HTTP*-Server gestartet werden, der das *Interface* für die Detektion bereitstellt. In Abbildung A.2 ist das Interface abgebildet.

Zuerst muss über das *Dropdown*-Menü ein Modell ausgewählt werden. Nachdem das Modell erfolgreich geladen worden ist, kann im Texteingabefeld der zu überprüfende Text eingegeben werden. Im unteren Teil des Interfaces gibt der Detektor nun das entsprechende Ergebnis aus.

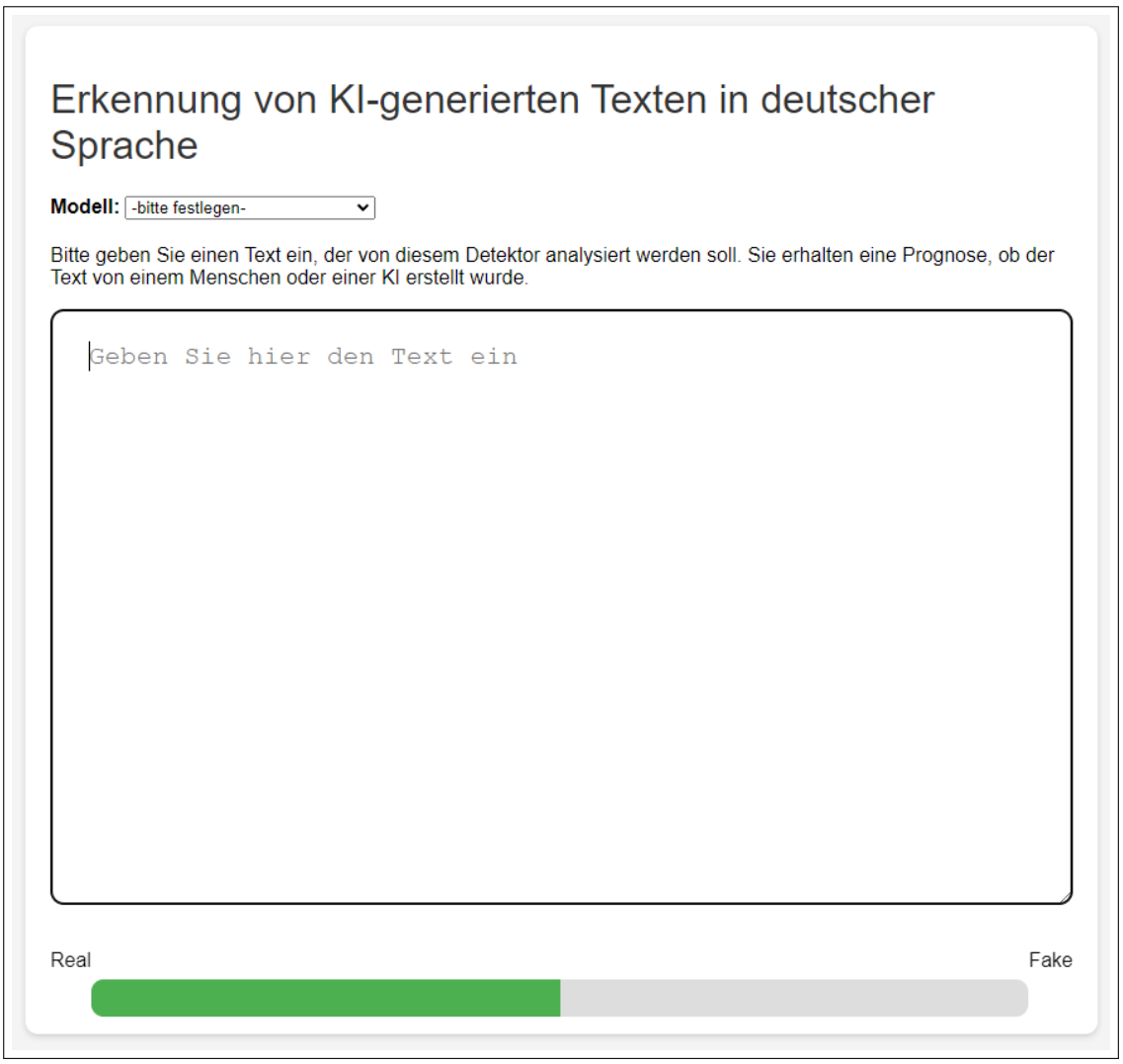

Abbildung A.2.: Interface

Lizenz

Das vorliegende Projekt, einschließlich des Quellcodes und der Daten, ist urheberrechtlich geschützt und Eigentum des Autors. Der Quellcode und die Daten werden ausschließlich zum Zweck der Bewertung der Masterarbeit des Autors zur Verfügung gestellt.

Jede Verwendung, Vervielfältigung, Verbreitung oder anderweitige Nutzung des Projekts, des Quellcodes oder der Daten außerhalb dieses Zwecks ist ohne die ausdrückliche schriftliche Zustimmung des Autors strengstens untersagt.

Durch den Zugriff auf den Quellcode und die Daten stimmen Sie den oben genannten Bedingungen zu und erkennen an, dass jegliche unerlaubte Nutzung eine Verletzung der Urheberrechte des Autors darstellt.

Literatur

Anthropic (11. Juli 2023), *[Claude] 2*, San Francisco (US), url: https://www.anthropic.com/index/claude-2 (besucht am 12. 08. 2023).

Aumiller, Dennis / Gertz, Michael (Juni 2022), „[Klexikon]: A German Dataset for Joint Summarization and Simplification“, in: *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, Marseille (FR): European Language Resources Association, S. 2693–2701, url: https://aclanthology.org/2022.lrec-1.288.

Baki, Shahryar u. a. (2017), „Scaling and effectiveness of email [masquerade] attacks: Exploiting natural language generation“, in: *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, S. 469–482.

Barrus, Tyler et al. (26. Apr. 2023), *[Pyspellchecker]*, Fredericksburg (US), url: https://pypi.org/project/language-tool-python/ (besucht am 23. 08. 2023).

Bhat, Shravya u. a. (2022), „Towards automated generation and evaluation of [questions] in educational domains“, in: *Proceedings of the 15th International Conference on Educational Data Mining*, Bd. 701.

Brown, Tom u. a. (2020), „Language models are [few-shot] learners“, in: *Advances in neural information processing systems* 33, S. 1877–1901.

Cai, Shuyang / Cui, Wanyun (2023), „[Evade] ChatGPT detectors via a single space“, in: *arXiv preprint arXiv:2307.02599*.

Chan, Branden / Schweter, Stefan / Möller, Timo (2020), „German's [next] language model“, in: *arXiv preprint arXiv:2010.10906*.

Chowdhery, Aakanksha u. a. (2022), „[PaLM]: Scaling language modeling with pathways“, in: *arXiv preprint arXiv:2204.02311*.

Coles, Cameron (19. Juni 2023), *ChatGPT vs. Bard: [Usage] statistics show Google trailing OpenAI*, Palo Alto (US), url: https://www.cyberhaven.com/blog/chatgpt-vs-google-bard-usage-statistics (besucht am 26. 08. 2023).

Cotton, Debby RE / Cotton, Peter A / Shipway, J Reuben (2023), „Chatting and cheating: Ensuring academic [integrity] in the era of ChatGPT“, in: *Innovations in Education and Teaching International*, S. 1–12.

Crothers, Evan / Japkowicz, Nathalie / Viktor, Herna L (2023), „Machine-generated Text: A Comprehensive Survey of Threat Models and Detection Methods“, in: *IEEE Access*.

Devlin, Jacob u. a. (2018), „[Bert]: Pre-training of deep bidirectional transformers for language understanding“, in: *arXiv preprint arXiv:1810.04805*.

Duarte, Fabio (13. Juli 2023), *Number of ChatGPT [Users]*, San Francisco (US), url: https://explodingtopics.com/blog/chatgpt-users (besucht am 26. 08. 2023).

Dwivedi, Yogesh K u. a. (2023), „“So what if ChatGPT wrote it?” Multidisciplinary [perspectives] on opportunities, challenges and implications of generative conversational AI for research, practice and policy“, in: *International Journal of Information Management* 71, S. 102642.

Elsen-Rooney (4. Jan. 2023), *[NYC] education department blocks ChatGPT on school devices, networks*, url: https://ny.chalkbeat.org/2023/1/3/23537987/nyc-schools-ban-chatgpt-writing-artificial-intelligence (besucht am 23. 07. 2023).

Fagni, Tiziano u. a. (2021), „[TweepFake]: About detecting deepfake tweets“, in: *Plos one* 16.5, e0251415.

Felder, Victor (31. Aug. 2022), *Confusable [Homoglyphs]*, Fredericksburg (US), url: `https://pypi.org/project/language-tool-python/` (besucht am 23. 08. 2023).

Fröhling, Leon / Zubiaga, Arkaitz (2021), „[Feature]-based detection of automated language models: tackling GPT-2, GPT-3 and Grover“, in: *PeerJ Computer Science* 7, Hannover, e443.

Gallé, Matthias u. a. (2021), „[Unsupervised] and distributional detection of machine-generated text“, in: *arXiv preprint arXiv:2111.02878*.

Gao, Catherine A u. a. (2023), „Comparing scientific [abstracts] generated by ChatGPT to real abstracts with detectors and blinded human reviewers“, in: *NPJ Digital Medicine* 6.1, S. 75.

Gao, Ji u. a. (2018), „Black-box generation of adversarial text sequences to [evade] deep learning classifiers“, in: *2018 IEEE Security and Privacy Workshops (SPW)*, IEEE, S. 50–56.

Garbe, Wolf et al. (24. Okt. 2022), *[Symspellpy]*, Fredericksburg (US), url: `https://pypi.org/project/language-tool-python/` (besucht am 23. 08. 2023).

Gehrmann, Sebastian / Strobelt, Hendrik / Rush, Alexander M (2019), „[GLTR]: Statistical detection and visualization of generated text“, in: *arXiv preprint arXiv:1906.04043*.

Goodfellow, Ian J / Shlens, Jonathon / Szegedy, Christian (2014), „Explaining and harnessing [adversarial] examples“, in: *arXiv preprint arXiv:1412.6572*.

Goyal, Shreya u. a. (2022), „A [survey] of adversarial defences and robustness in nlp“, in: *ACM Computing Surveys*.

Guo, Biyang u. a. (2023), „How [close] is chatgpt to human experts? comparison corpus, evaluation, and detection“, in: *arXiv preprint arXiv:2301.07597*.

Hacker, Philipp / Engel, Andreas / Mauer, Marco (2023), „[Regulating] ChatGPT and other large generative AI models“, in: *Proceedings of*

the 2023 ACM Conference on Fairness, Accountability, and Transparency, S. 1112–1123.

Hawking, Stephen (19. Okt. 2016), *The best or worst thing to happen to [humanity]*, url: `https://www.beispielwebseite.com/rede-transkript` (besucht am 22. 07. 2023).

Ippolito, Daphne u. a. (2020), „[Automatic] detection of generated text is easiest when humans are fooled“, in: *arXiv preprint arXiv:1911.00650*.

Islam, Niful u. a. (2023), „[Distinguishing] Human Generated Text From ChatGPT Generated Text Using Machine Learning“, in: *arXiv preprint arXiv:2306.01761*.

Jawahar, Ganesh / Abdul-Mageed, Muhammad / Lakshmanan, Laks VS (2020), „[Automatic] detection of machine generated text: A critical survey“, in: *arXiv preprint arXiv:2011.01314*.

Jeblick, Katharina u. a. (2022), „Chatgpt makes [medicine] easy to swallow: An exploratory case study on simplified radiology reports“, in: *arXiv preprint arXiv:2212.14882*.

Jia, Robin / Liang, Percy (2017), „[Adversarial] examples for evaluating reading comprehension systems“, in: *arXiv preprint arXiv:1707.07328*.

Kasneci, Enkelejda u. a. (2023), „ChatGPT for good? On opportunities and challenges of large language models for [education]“, in: *Learning and Individual Differences* 103, S. 102274.

Keung, Phillip u. a. (2020), „The Multilingual Amazon [Reviews] Corpus“, in: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

Khalil, Mohammad / Er, Erkan (2023), „[Will] ChatGPT get you caught? Rethinking of plagiarism detection“, in: *arXiv preprint arXiv:2302.04335*.

Kratzke, Nane (3. Mai 2020), *Monthly Samples of German [Tweets]*, Genf (CH), url: `https://zenodo.org/record/3783478` (besucht am 12. 08. 2023).

Krawczyk, Jack (13. Juli 2023), *Produktivität und Kreativität steigern: Bard jetzt auch in [Deutschland] verfügbar*, San Francisco (US), url: https://blog.google/intl/de-de/unternehmen/technologie/bard-deutschland-verfuegbarkeit-ki/ (besucht am 12. 08. 2023).

Krishna, Kalpesh u. a. (2023), „[Paraphrasing] evades detectors of ai-generated text, but retrieval is an effective defense“, in: *arXiv preprint arXiv:2303.13408*.

Lample, Guillaume / Conneau, Alexis (2019), „Cross-lingual language model pretraining“, in: *arXiv preprint arXiv:1901.07291*.

Li, Jinfeng u. a. (2018), „[Textbugger]: Generating adversarial text against real-world applications“, in: *arXiv preprint arXiv:1812.05271*.

Li, Linyang u. a. (2023), „[Origin] Tracing and Detecting of LLMs“, in: *arXiv preprint arXiv:2304.14072*.

Liang, Weixin u. a. (2023), „GPT [detectors] are biased against non-native English writers“, in: *arXiv preprint arXiv:2304.02819*.

Liu, Zhengxiao u. a. (2020), „De-co: A two-step [spelling] correction model for combating adversarial typos“, in: *2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, IEEE, S. 554–561.

Ma, Yongqiang u. a. (o. D.), *[AI] vs. Human–Differentiation Analysis of Scientific Content Generation*.

Mi, Steven (2020), *German [News] Dataset*, doi: 10.34740/KAGGLE/DSV/1679751, url: https://www.kaggle.com/dsv/1679751 (besucht am 12. 08. 2023).

Min, Bonan adn others (2021), „Recent advances in natural language processing via large pre-trained language [models]: A survey“, in: *ACM Computing Surveys*.

Minixhofer, Benjamin / Paischer, Fabian / Rekabsaz, Navid (2021), „[WECHSEL]: Effective initialization of subword embeddings for cross-lingual transfer of monolingual language models“, in: Seattle (US).

Mitchell, Eric u. a. (2023), „[Detectgpt]: Zero-shot machine-generated text detection using probability curvature“, in: *arXiv preprint arXiv:2301.11305*, Stanford (US).

Mitrović, Sandra / Andreoletti, Davide / Ayoub, Omran (2023), „[Chatgpt] or human? detect and explain. explaining decisions of machine learning model for detecting short chatgpt-generated text“, in: *arXiv preprint arXiv:2301.13852*.

Möller, Timo / Risch, Julian / Pietsch, Malte (2021), „[GermanQuAD] and GermanDPR: Improving non-English question answering and passage retrieval“, in: *arXiv preprint arXiv:2104.12741*.

Myint, Steven et al. (18. Apr. 2022), *A grammar [checker] for Python*, Fredericksburg (US), url: `https://pypi.org/project/language-tool-python/` (besucht am 23. 08. 2023).

o.V. (27. Feb. 2021), *German language reviews of [doctors] by patients 2021*, Austin (US), url: `https://data.world/mc51/german-language-reviews-of-doctors-by-patients-2021` (besucht am 12. 08. 2023).

o.V. (5. Dez. 2022), *Generative AI (e.g., ChatGPT) is [banned]*, url: `https://meta.stackoverflow.com/questions/421831/temporary-policy-generative-ai-e-g-chatgpt-is-banned` (besucht am 22. 07. 2023).

OpenAI (2019a), *Giant [Language] model Test Room*, Havard (US), url: `http://gltr.io/dist/` (besucht am 17. 08. 2023).

OpenAI (3. Mai 2019b), *GPT-2 [output] dataset*, San Francisco (US), url: `https://github.com/openai/gpt-2/blob/master/domains.txt` (besucht am 27. 07. 2023).

OpenAI (30. Nov. 2022), *Introducing [ChatGPT]*, San Francisco (US), url: `https://openai.com/blog/chatgpt` (besucht am 19. 07. 2023).

OpenAI (13. März 2023a), *[GPT-4] is OpenAI's most advanced system, producing safer and more useful responses*, San Francisco (US), url: https://openai.com/gpt-4 (besucht am 12. 08. 2023).

OpenAI (16. Aug. 2023b), *[Tokenizer]*, San Francisco (US), url: https://platform.openai.com/tokenizer (besucht am 16. 08. 2023).

OpenAI (31. Jan. 2023c), *New AI [classifier] for indicating AI-written text*, San Francisco (US), url: https://openai.com/blog/new-ai-classifier-for-indicating-ai-written-text (besucht am 01. 08. 2023).

Ouyang, Long u. a. (2022), „Training language models to follow instructions with human [feedback]", in: *Advances in Neural Information Processing Systems* 35, S. 27730–27744.

Pegoraro, Alessandro u. a. (2023), „To [ChatGPT], or not to ChatGPT: That is the question!", in: *arXiv preprint arXiv:2304.01487*, Darmstadt.

Pichai, Sundar (6. Feb. 2023), *An important [next] step on our AI journey*, San Francisco (US), url: https://blog.google/technology/ai/bard-google-ai-search-updates/ (besucht am 12. 08. 2023).

Pires, Telmo / Schlinger, Eva / Garrette, Dan (2019), „How [multilingual] is multilingual BERT?", in: *arXiv preprint arXiv:1906.01502*, Multilingual.

Pu, Jiameng u. a. (2022), „[Deepfake] text detection: Limitations and opportunities", in: *2023 IEEE Symposium on Security and Privacy (SP)*, IEEE, S. 1613–1630.

Radford, Alec u. a. (2018), „Improving language understanding by [generative] pre-training", in.

Rudolph, Jürgen / Tan, Samson / Tan, Shannon (2023a), „[ChatGPT]: Bullshit spewer or the end of traditional assessments in higher education?", in: *Journal of Applied Learning and Teaching* 6.1.

Rudolph, Jürgen / Tan, Shannon / Tan, Samson (2023b), „War of the [chatbots]: Bard, Bing Chat, ChatGPT, Ernie and beyond. The new AI gold

rush and its impact on higher education“, in: *Journal of Applied Learning and Teaching* 6.1.

Sadasivan, Vinu Sankar u. a. (2023), „Can [ai-generated] text be reliably detected?“, in: *arXiv preprint arXiv:2303.11156*.

Sadeghi, McKenzie u. a. (17. Juli 2023), *Tracking AI-enabled [Misinformation]*, url: `https://www.newsguardtech.com/special-reports/ai-tracking-center/` (besucht am 23. 07. 2023).

Scheible, Raphael u. a. (2020), „[Gottbert]: a pure german language model“, in: *arXiv preprint arXiv:2012.02110*.

Schmid, Katharina (12. Dez. 2019), *DBMDZ [German] BERT models*, New York City (US), url: `https://huggingface.co/dbmdz/bert-base-german-cased` (besucht am 24. 08. 2023).

Solaiman u. a. (2019), „Release strategies and the social [impacts] of language models“, in: *arXiv preprint arXiv:1908.09203*.

Susnjak, Teo (2022), „ChatGPT: The end of online [exam] integrity?“, in: *arXiv preprint arXiv:2212.09292*.

Tang, Ruixiang / Chuang, Yu-Neng / Hu, Xia (2023), „The [science] of detecting llm-generated texts“, in: *arXiv preprint arXiv:2303.07205*.

Tian, Edward (2023), *[GPTZero]*, Princeton (US), url: `https://gptzero.me/` (besucht am 01. 08. 2023).

Touvron, Hugo u. a. (2023), „[LLaMA] 2: Open Foundation and Fine-Tuned Chat Models“, in: *Journal of Applied Learning and Teaching* 6.1.

Uchendu, Adaku u. a. (2020), „[Authorship] attribution for neural text generation“, in: *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, University Park (US), S. 8384–8395.

Vaswani, Ashish u. a. (2017), „[Attention] is all you need“, in: *Advances in neural information processing systems* 30.

Wei, Jason u. a. (2022), „[Emergent] abilities of large language models“, in: *arXiv preprint arXiv:2206.07682*.

Wolff, Max / Wolff, Stuart (2020), „[Attacking] neural text detectors“, in: *arXiv preprint arXiv:2002.11768*.

Yang, Jingfeng u. a. (2023), „Harnessing the power of llms in practice: A [survey] on chatgpt and beyond“, in: *arXiv preprint arXiv:2304.13712*.

Zellers, Rowan u. a. (2019), „[Defending] against neural fake news“, in: *Advances in neural information processing systems* 32, Washington (US).

Zeng, Jiehang u. a. (2021), „Certified robustness to text adversarial attacks by randomized [mask]“, in: *Computational Linguistics* 49.2, S. 395–427.

Zhao, Wayne Xin u. a. (2023), „A survey of large [language] models“, in: *arXiv preprint arXiv:2303.18223*, url: `https://arxiv.org/abs/2303.18223`.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Ort, Datum Tom Tlok