Pröhl, Thorsten; Mohrhardt, Radoslaw; Förster, Niels; Putzier, Erik; Zarnekow, Rüdiger

Article — Published Version

Erkennungsverfahren für KI-generierte Texte: Überblick und Architekturentwurf

HMD Praxis der Wirtschaftsinformatik

Provided in Cooperation with:
Springer Nature

Suggested Citation: Pröhl, Thorsten; Mohrhardt, Radoslaw; Förster, Niels; Putzier, Erik; Zarnekow, Rüdiger (2024) : Erkennungsverfahren für KI-generierte Texte: Überblick und Architekturentwurf, HMD Praxis der Wirtschaftsinformatik, ISSN 2198-2775, Springer Fachmedien Wiesbaden GmbH, Wiesbaden, Vol. 61, Iss. 2, pp. 418-435, https://doi.org/10.1365/s40702-024-01051-w

This Version is available at:
https://hdl.handle.net/10419/315879

HMD Praxis der Wirtschaftsinformatik (2024) 61:418–435
https://doi.org/10.1365/s40702-024-01051-w

SCHWERPUNKT

Erkennungsverfahren für KI-generierte Texte: Überblick und Architekturentwurf

Thorsten Pröhl · Radoslaw Mohrhardt · Niels Förster · Erik Putzier · Rüdiger Zarnekow

Eingegangen: 31. Oktober 2023 / Angenommen: 25. Januar 2024 / Online publiziert: 19. Februar 2024

Zusammenfassung Durch Transformer-basierte KI-Systeme wurden große Fortschritte, u. a. in den Bereichen Textverarbeitung und -verständnis, erreicht. Diese Deep-Learning-Modelle ermöglichen das Generieren von Texten und bilden die Grundlage moderner Sprachmodelle. Die rasante Entwicklung der letzten Jahre hat große Sprachmodelle, wie ChatGPT, Bard oder VICUNA-13B, hervorgebracht.

Der Beitrag präsentiert die Entwicklung der Sprachmodelle hin zu den großen Sprachmodellen. Durch die fortschreitende Entwicklung der Sprachmodelle ergeben sich vielfältige Möglichkeiten und Probleme, weshalb eine Erkennung von LLM-generierten Texten wichtig ist. Dieser Artikel stellt unterschiedliche Ansätze bekannter Erkennungsverfahren dar. Neben statistischen Klassifizierungsverfahren werden auch Deep-Learning-basierte und Zero-Shot-Verfahren besprochen. Daneben werden ein kompressionsorientierter Ansatz vorgestellt sowie Kennzeichnungsverfahren präsentiert. Nach dem tabellarischen Vergleich der in der Literatur vorgestellten Verfahren werden implementierte Softwaredetektoren präsentiert. Im Anschluss werden Überlegungen zum Entwurf eines Trainingsdatensatzes aufgezeigt, wodurch die Grundlage für einen eigenen Ansatz zur Erkennung von KI-generierten Texten in deutscher Sprache geschaffen wird. Darüber hinaus werden die Architektur und das Design des eigenen Ansatzes, dem KI-Inhalte-Detektor, vorgestellt und beschrieben sowie die Limitationen aufgezeigt.

Schlüsselwörter Künstliche Intelligenz · Verarbeitung natürlicher Sprache · Große Sprachmodelle · KI-Erkennung · Erkennung von Sprachmodellen

✉ Thorsten Pröhl · Erik Putzier · Rüdiger Zarnekow
Fachgebiet Informations- und Kommunikationsmanagement, Technische Universität Berlin, Straße des 17. Juni 135, 10623 Berlin, Deutschland
E-Mail: t.proehl@tu-berlin.de

Thorsten Pröhl · Radoslaw Mohrhardt · Niels Förster · Erik Putzier
Trustami GmbH, Bamberger Straße 40, 10779 Berlin, Deutschland

Detection Methods for AI-generated Texts: Overview and Architectural Design

Abstract Transformer-based AI systems have made great progress in areas such as text processing and comprehension. These deep learning models enable the generation of texts and form the basis of modern language models. The rapid development of recent years has produced large language models such as ChatGPT, Bard and VICUNA-13B.

This article presents the development of language models towards large language models. The progressive development of language models has resulted in unimagined possibilities and, at the same time, a variety of problems, which is why it is important to recognize LLM-generated textual content. This article presents the different approaches of known recognition methods. In addition to statistical classification methods, deep learning-based and zero-shot methods are also discussed. In addition, a compression-oriented approach is presented as well as labeling methods. After a tabular comparison of the methods presented in the literature, implemented software detectors are presented. Subsequently, considerations for the design of a training dataset are presented, creating the basis for an own approach for the recognition of AI-generated texts in German. In addition, the architecture and design of our own approach, the AI content detector, is presented and described and the limitations are shown.

Keywords Artificial Intelligence · Natural Language Processing · Large Language Model Detection · AI Detection · Detection of AI-Generated Text · AI Content Detection

1 Entwicklung der generativen Sprachmodelle

In den letzten Jahren hat die Forschung im Bereich Künstliche Intelligenz (KI) zahlreiche Fortschritte erzielt. Neben KIs, die Bilder und Kunst erschaffen, wie Midjourney oder DALL·E 2, hat eine erhebliche Entwicklung im Bereich der Sprachmodelle stattgefunden. Gerade den generativen Sprachmodellen wird viel Aufmerksamkeit zuteil. Generative Sprachmodelle sind KI-Systeme, die das Ziel haben, Texte zu generieren. Dabei sollen die Texte natürlich klingen und den Eindruck vermitteln, dass ein Mensch sie verfasst hat. Diese Form von spezialisierten KI-Systemen erlernen im Training auf Basis einer riesigen Datenmenge Muster der jeweiligen Sprache und reagieren während der Modellinferenz auf Eingaben (Prompts).

Die heutigen generativen Sprachmodelle oder auch großen Sprachmodelle (engl. Large Language Models, LLM) basieren auf zahlreichen Vorarbeiten im Bereich der Verarbeitung natürlicher Sprachen (engl. Natural Language Processing, NLP) und sind durch den Fortschritt der KI-Algorithmen (Deep Learning, Transfer Learning), durch die Verfügbarkeit moderner Grafikkarten (engl. Graphics Processing Unit, GPU) und hochspezialisierter Tensor Prozessoren (engl. Tensor Processing Unit, TPU) sowie durch stark verteilte Lernverfahren möglich geworden. Weiterhin wer-

den die genannten Vorarbeiten und Fortschritte auf riesige Datenmengen angewandt, um Sprachmuster zu erlernen (Sarker 2022).

In den 50er-Jahren wurden erste Computersysteme entwickelt, welche aus Regeln und statistischen Methoden einfache Sätze bilden konnten. Im Georgetown-Experiment wurden für nachrichtendienstliche Zwecke russische Texte ins Englische übersetzt (Gordin 2016), in den 60er-Jahren wurden erste Chatbots, wie ELIZA, von Weizenbaum (1966) implementiert.

Als einer der Vorreiter in der Entwicklung der modernen künstlichen Intelligenz kann Frank Rosenblatt gesehen werden (Tappert 2019). Rosenblatt (1958) entwickelte das Modell des Perzeptrons, das als künstliches Neuron betrachtet werden kann und als Grundbaustein moderner künstlicher Intelligenz gilt.

Infolge dieser weitreichenden Veröffentlichung kam es jedoch in den 70er-Jahren zum ersten sogenannten KI-Winter, einer Phase, in der die Hoffnung an das Potenzial künstlicher Intelligenz zurückging und es zu Kapitalabzug und einem Rückgang in der Forschung kam. Der Zeitraum um die Jahrtausendwende wird als weiterer KI-Winter betrachtet (Harguess und Ward 2022). In den 80er-Jahren wurden neue Ansätze für die Textgenerierung entwickelt, welche auf Basis vom Hidden Markov Model (HMM) funktionieren (Manning und Schütze 2005). Wahrscheinlichkeitsverteilungen bilden bei diesen Modellen die Grundlage und bieten damit eine deutlich verbesserte Modellierung von Wortzusammenhängen in Sätzen sowie Texten.

In den 2000er Jahren sind Künstliche Neuronale Netzte (KNN, engl. Artificial Neural Network, ANN), die maßgeblich auf Rosenblatt (1958) aufbauen, und Deep Learning (DL) in den Fokus der Industrie gerückt, weil sich die Technik nun entsprechend weiterentwickelt hat. Künstliche Neuronale Netze sind Algorithmen, die die Funktionsweise menschlicher Gehirne imitieren. Es handelt sich um Netzwerke, die über eine große Anzahl von Knoten (künstliche Neuronen) Eingangsdaten verarbeiten und in spezifischer Form, je nach Konstruktion des Netzwerks, wieder ausgeben.

Komplexere künstliche neuronale Netze haben mehrere Ebenen, die zwischen der Eingangs- und der Ausgangsebene liegen (sogenannte Hidden Layers). Je größer die Anzahl der Ebenen ist, desto tiefer ist das Netzwerk, wodurch der Begriff Deep Learning resultiert. Im Vergleich zu anderen Algorithmen des maschinellen Lernens sind KNNs in der Lage komplexe, nicht-lineare Zusammenhänge innerhalb der Daten zu erlernen (Alzubaidi et al. 2021). Im Bereich NLP hat sich die Entwicklung von Recurrent Neural Networks (RNNs) und Long Short-Term Memory (LSTM) Modellen, einer Weiterentwicklung der RNNs, als großer Fortschritt erwiesen, da sie in der Lage sind die Interdependenzen von Wörtern innerhalb eines Satzes zu erlernen (Sundermeyer et al. 2012).

Diesen Fortschritten folgten die sogenannten Transformer-Architekturen (Vaswani et al. 2017). Ein zentrales Konzept, welches im Rahmen dieser Architekturen vorgestellt wurde, sind die Attention-Mechanismen (Vaswani et al. 2017). Durch den Attention-Mechanismus, der über eine zusätzliche Ebene in KNNs integriert wird, werden diese in die Lage versetzt den Kontext jedes einzelnen Wortes eines Satz oder Textes, unabhängig von der Länge der Eingangssequenz, zu berücksichtigen und diesen nicht wieder zu verlernen, was ein Vorteil gegenüber RNNs ist. Der Transformer wird in die Lage versetzt, sehr weitreichende Interdependenzen

zwischen Wörtern, Sätzen und innerhalb von Texten zu erkennen, woraus sich das Potenzial eines tiefen Textverständnisses ergibt, dass die Basis für qualitativ hochwertige maschinell generierte Texte darstellt.

Devlin et al. (2018) stellen etwa in ihrem Beitrag das BERT (Bidirectional Encoder Representations from Transformers) Modell vor, das die Grundlage für viele weitere Transformer-Varianten und Weiterentwicklungen, wie RoBERTa (Liu et al. 2019) oder German BERT (Chan et al. 2020), darstellt.

Im Jahr 2020 präsentierte OpenAI den Generative Pre-trained Transformer 3 (GPT-3). GPT-3 nutzt 175 Mrd. Parameter und stellt damit eines der leistungsfähigsten Sprachmodelle zur Generierung von Texten dar. Dieses Modell generiert natürlich klingende Texte auf einem sehr hohen sprachlichen Niveau und eignet sich damit für unterschiedliche Aufgaben, wie Zusammenfassungen zu schreiben, Übersetzungen anzufertigen oder für den Einsatz als Chatbot.

In den folgenden Jahren hat OpenAI die Weiterentwicklungen GPT-3.5 und GPT-4 vorgestellt. Dabei steigen zum einen die Anzahl der Parameter sehr stark an und zum anderen werden Trainingsdaten aus weiteren Quellen verwendet. GPT-4 ist nicht mehr auf Texte beschränkt, sondern akzeptiert auch Bilder als Input (multimodal). Weiterhin werden die neusten GPT-Modelle von OpenAI kontinuierlich weiter trainiert und mit aktuellen Trainingsdaten versorgt, so dass u. a. GPT-4 zu aktuellen Ereignissen oder Geschehnissen passende Texte generieren oder Fragen beantworten kann.

Parallel zu der Entwicklung der GPT-Modelle von OpenAI veröffentlichte Google das Bard Modell, das ebenfalls kontinuierlich mit neuen Trainingsdaten versorgt wird und damit auch auf aktuelle Geschehnisse richtig reagieren kann.

Darüber hinaus entwickelte die Open Source Community kleine hochspezialisierte und sehr große Sprachmodelle, die in ihrer Performance bereits mit GPT-3.5 vergleichbar sind, wie z. B. das VICUNA-13B Modell (Vicuna 2023).

2 Resultierende Probleme

Der Einsatz von generativen Sprachmodellen birgt nicht nur Potenziale, sondern auch verschiedenartige Herausforderungen und Probleme, welche im Rahmen einer Literaturanalyse identifiziert worden sind.

Die folgenden Bereiche bedürfen besonderer Aufmerksamkeit:

- *Fake News und Fehlinformationen*: Sprachmodelle können für das automatische und schnelle Erstellen von Texten mit kontrafaktischen oder fiktionalen Inhalten verwendet werden. Fake News und Fehlinformationen können die öffentliche Meinung beeinflussen und dafür sorgen, dass das Vertrauen in Informationen gestört wird (Su et al. 2023).
- *Diskriminierung und Bias*: Die Grundlage für das Training von LLMs stellen große Mengen von Texten dar. Innerhalb dieser Trainingsdaten können historische Vorurteile, spezifische Meinungen oder Gesinnungen kodiert sein, die wiederum bei der Generierung in die neuen Texte einfließen (Venkit et al. 2023).

- *Spam und Phishing*: Für die Erstellung von Inhalten, die eine betrügerische Absicht verfolgen, wie Phishing-E-Mails oder Phishing-Kurznachrichten, können Sprachmodelle genutzt werden (Hazell 2023).
- *Manipulierende Anfragen*: Sprachmodelle können, je nach Modell und Betriebsmodus, auch Anfragen annehmen und diese für das eigene Lernen verwenden. Das sich kontinuierlich trainierende Sprachmodell kann auf diese Weise nachhaltig negativ verändert werden, wodurch es möglicherweise unangemessene und ethisch nicht korrekte Antworten auf menschliche Eingaben liefert (Peng et al. 2023).
- *Identitätsdiebstahl*: Mit der Hilfe von LLMs lassen sich Texte erstellen, um andere Personen zu imitieren und deren Identität zu stehlen (Hazell 2023).
- *Ratschläge mit gesundheitlichen Auswirkungen*: Sprachmodelle können den Eindruck erwecken, dass sie eine große Fachkompetenz aufweisen, z. B. wie ein Arzt klingen und gesundheitsrelevante Ratschläge geben. Diese Ratschläge können irreführend oder gesundheitsgefährdend sein, wenn das LLM die falschen Informationen für die Beantwortung der Fragen verwendet oder durch gezielte Anfragen manipuliert wurde (Haupt und Marks 2023).
- *Bewertung von Eigenleistungen im Bildungswesen*: Lehrkräfte können bei Aufsätzen, Hausarbeiten oder Abschlussarbeiten Probleme bei der Bewertung der Studierenden bekommen, weil sie nur schwerlich unterscheiden können, wer die jeweilige Arbeit verfasst hat: ein Mensch oder ein LLM (Orenstrakh et al. 2023).
- *Schutz von Privatsphäre:* Der Schutz der Privatsphäre von Personen, allgemeiner Datenschutz oder Urheberrecht können in der Ausgabe verletzt werden (Yao et al. 2023).

Erkennungsverfahren, die eine Überprüfung, ob ein Text KI-generiert ist, ermöglichen, bilden eine zentrale Grundlage im Umgang mit den genannten Herausforderungen und Problemen.

3 Überblick zu Erkennungsverfahren

Zur Erkennung von KI-generierten Texten werden in der Forschung diverse Ansätze und Methoden diskutiert. Mit Hilfe einer Literaturanalyse werden die folgenden Verfahren identifiziert, die sich in fünf Bereiche einteilen lassen. Neben statistischen Klassifizierungsverfahren werden auch Deep Learning basierte und Zero-Shot-Verfahren präsentiert. Kompressionsorientierte Verfahren stellen einen Spezialfall dar. Zum Schluss werden Verfahren zur Kennzeichnung von KI-generierten Texten vorgestellt, wobei auch Umgehungsverfahren diskutiert werden.

3.1 Statistische Klassifizierungsverfahren

Im klassischen Maschinellen Lernen (ML), werden als Input Variablen sogenannte Features benötigt, die im Kontext von Erkennungsverfahren dafür genutzt werden, um zu klassifizieren, ob ein Text maschinell oder von einem Menschen erstellt worden ist. Die Features werden für die Texte der Trainingsdaten berechnet, die sowohl KI-generierte als auch von Menschen geschriebene Texte beinhalten. Im klassischen

Maschinellen Lernen handelt es sich bei den Features immer um Zahlenwerte, die benötigt werden, da die benutzten Modelle, wie z. B. Random Forest, nicht in der Lage sind Texte unmittelbar zu verarbeiten. Features können als quantitative Charakteristika eines Textes interpretiert werden, die es den Modellen ermöglichen Unterscheidungsgrenzen zu erlernen. Die Features können beispielsweise die Anzahl der Wörter eines Textes sein, aber auch eine größere Komplexität aufweisen, wie zum Beispiel die Anzahl von Superlativen im Verhältnis zur Anzahl der Adjektive eines Textes (Shuai et al. 2022). Im Zuge der Feature Extraktion (engl. Feature Extraction), sollten möglichst viele und unterschiedliche Features konstruiert werden, was einen größeren Entwicklungsaufwand erfordert. Tang et al. (2023) nennen in ihrem Artikel unter anderem die Wort-Diversität (Guo et al. 2023), das Sentiment, die allgemeine Lesbarkeit und die Absichtslosigkeit oder Neutralität eines Textes (Fröhling und Zubiaga 2021), anhand derer sich KI-generierte Texte identifizieren lassen. Letztere zeigen, dass die Auswahl des Trainingsdatensatzes und der konstruierten Features einen sehr großen Einfluss auf den resultierenden Klassifikator und dessen Generalisierbarkeit haben.

Außerdem wird festgestellt, dass Ensemble-Klassifikatoren (Kombination mehrerer unterschiedlicher Klassifikatoren) mit hochwertigen Features konkurrenzfähig gegenüber rechenintensiveren LLM-Detektoren sind und als „erste Verteidigungslinie“ bei der Identifikation maschinell generierter Texte betrachtet werden können.

Festgehalten wird aber auch, dass aufgrund der fortschreitenden Entwicklung der LLMs und der einfachen Anpassungsmöglichkeit der generierten Texte über die Prompt Eingabe schnell auf derartige Muster reagiert werden könnte, sodass vermeintlich gefundene Diskriminationsmerkmale nur eine geringe Robustheit aufweisen könnten.

Eine im Kontext der Sprachmodellerkennung verwendete Kennzahl ist die sogenannte „Burstiness“, die beschreibt, wie verschieden die Sätze eines Textes sind. Ein menschlich geschriebener Text variiert stärker als ein Sprachmodelltext in der Satzlänge und einzelne selten vorkommende Wörter werden häufig in kurzen Abständen mehrfach verwendet und danach nicht mehr. Da Sprachmodelle in ihrer Ursprungsform darauf trainiert werden zu bestimmen, den passenden Token (Textbausteine) als nächsten Token einzusetzen, neigen sie dazu, häufig vorkommende Wörter auszuwählen und durchschnittliche Satzlängen zu bilden. Intuitiver lässt sich feststellen, dass Menschen unterschiedliche Schreibstile bezüglich verwendeter Wörter, Satzlänge oder grammatikalischer Strukturen haben. Sprachmodelle hingegen imitieren den durchschnittlichen Schreibstil aller Inhalte, auf denen sie trainiert wurden. Diese Diskrepanz soll durch die Bestimmung der Burstiness erkannt werden. Allerdings fällt die Aussagekraft dieser Kennzahl, wenn die Leistungsfähigkeit der LLMs zunimmt (Chakraborty et al. 2023a).

Eine weitere geeignete Kennzahl ist die „Perplexity“ (Chen et al. 1998), mit der die Wahrscheinlichkeit bestimmt werden kann, dass eine Sequenz von einem bestimmten Modell generiert wurde. Dafür wird für jeden Token die Wahrscheinlichkeit bestimmt, dass er als nächster Token generiert wird. Sind die Wahrscheinlichkeiten jeweils hoch, ist das Modell weniger „perplex“ diese Sequenz zu sehen, was auf einen generierten Text hinweist. Analog dazu bedeuten niedrige Wahrscheinlichkeiten eine hohe Perplexity und einen wahrscheinlich nicht von dem Modell

generierten Text. Angenommen, ein zu überprüfender Text wurde von einem LLM generiert. Wenn dieser Text für jeden Token als Input des LLMs genommen wird, um die Wahrscheinlichkeiten für den nächsten Token ausgeben zu lassen, werden die tatsächlichen Tokens mit einer hohen Wahrscheinlichkeit vorgeschlagen. Bei einem menschlich generierten Text besteht dieser Zusammenhang nicht, weswegen die Perplexity ein hilfreicher Indikator für die LLM-Erkennung sein kann.

Sowohl Burstiness als auch Perplexity werden unter anderem von GPTZero, einem Programm zur Sprachmodellerkennung, verwendet. Es handelt sich bei beiden Kennzahlen um komplexe Features eines Textes, die neben den zu Beginn des Kapitels genannten Features als Input Variablen für Klassifikatoren benutzt werden können.

3.2 Deep-Learning-Klassifizierungsverfahren

Bei Deep-Learning-Verfahren wird ein künstliches neuronales Netz auf einem Datensatz von LLM-generierten und von Menschen geschriebenen Texten trainiert, wodurch das Modell die Fähigkeit erlangt, auch zu neuen Inputtexten eine Einschätzung bezüglich KI-Generierung zu treffen. Eine Implementierung eines solchen Verfahrens ist der von OpenAI veröffentlichte Klassifikator (Kirchner et al. 2023).

Um ein solches Modell zu trainieren, wird eine große Menge an Texten, die jeweils als KI-generiert und nicht KI-generiert gelabelt werden, erstellt. Dieser Datensatz wird dann für das Training des neuronalen Netzes verwendet. Dabei werden keine Features festgelegt, anhand jener das Modell eine Einschätzung vornimmt, sondern die Features werden im Laufe des Trainingsprozess selbst durch das neuronale Netz erlernt. Mit Hilfe der gelernten Zusammenhänge ist es dem Modell dann möglich, in der Inferenzphase für neue Texte eine Entscheidung zu treffen, ob diese von einer KI generiert sind oder nicht.

Der von OpenAI Anfang 2023 veröffentlichte Klassifikator wurde aufgrund mangelnder Präzision bereits im Juli desselben Jahres vorerst eingestellt. Der Hauptgrund für die schlechte Leistung ist die hohe Abhängigkeit von den Trainingsdaten. Neuronale Netze lassen sich nur schlecht auf Inputsequenzen anwenden, die nicht in einer ähnlichen Form in den Trainingsdaten vorzufinden sind. Da die Inputsequenzen für ein allgemeines LLM-Erkennungsmodell aber sowohl inhaltlich als auch in der vorliegenden Sprache oder dem linguistischen Stil äußerst divers sind, würde ein enormer Datensatz benötigt werden, um all diese Variationen zu berücksichtigen. Denkbar ist eine Verwendung eines Deep Learning Ansatzes hingegen, um eine LLM-Erkennung bei einer spezifischen Gruppe von Texten vorzunehmen. Es wäre beispielsweise möglich ein Transformer Modell, mit KI- und nicht KI-generierten Texten, zu einem gemeinsamen Thema und ähnlicher Modalität, z. B. einem Essay, zu trainieren, weil dadurch die Voraussetzung vielseitiger Trainingsdaten entfällt. Diese Spezialisierungen umgehen die inhärente Schwäche der schlechten Generalisierbarkeit von Deep-Learning Modellen und ermöglichen eine effektive Verwendung dieser zur LLM-Texterkennung.

3.3 Zero-Shot-Verfahren

Namensgebend für Zero-Shot-Ansätze ist, dass kein vorhergehendes Modelltraining und die damit verbundenen Datensätze benötigt werden. Hierbei wird lediglich der zu überprüfende Text und die Sprachmodell-Schnittstelle verwendet. Dadurch erübrigt sich die aufwendige Suche nach geeigneten Trainingsdatensätzen.

Ein aktuelles Beispiel für die Implementierung eines Zero-Shot-Ansatzes ist DetectGPT von Mitchell et al. (2023). Bei diesem Ansatz werden zunächst Perturbationen des zu überprüfenden Textes generiert, wobei einzelne Wörter oder Wortgruppen durch semantisch korrekte Alternativen ersetzt werden. Dies wird durch die Verwendung eines weiteren LLMs, in diesem Fall T5 (Raffel et al. 2019) realisiert. Anschließend wird durch Zugriff auf die API des zu überprüfenden Modells für jeden Text die logarithmierte Wahrscheinlichkeit dafür bestimmt, dass die Sequenz von diesem Sprachmodell generiert wurde. Diese Wahrscheinlichkeit basiert auf dem Durchschnitt der Wahrscheinlichkeiten für jeden Token im Text.

Für die Klassifikation des Textes ist die Anordnung der zuvor ermittelten Wahrscheinlichkeitswerte entscheidend. Wenn der ursprüngliche Text tatsächlich von dem Sprachmodell generiert wurde, ist der Wahrscheinlichkeitswert für diese Generierung wesentlich höher als der Durchschnitt der Wahrscheinlichkeiten für die Perturbationen. Im Gegensatz dazu weichen bei Texten, die nicht von dem Sprachmodell generiert wurden, die Wahrscheinlichkeitswerte für den Ursprungstext und den Durchschnitt der Perturbationen nur geringfügig voneinander ab. Wenn beispielsweise ein modellgenerierter Text vorliegt, würden zuerst Perturbationen generiert werden. Daraufhin würden für den Originaltext und die Perturbationen, ähnlich wie bei der Bestimmung der Perplexity, die Wahrscheinlichkeiten für jeden Token bestimmt werden. Dabei würde die durchschnittliche Wahrscheinlichkeit für den Originaltext höher sein, weil die Tokens des Originaltexts eher die sind, die das Modell wieder einsetzen würde, als die Tokens der generierten Perturbationen. Anhand des Ausmaßes der Abweichung zwischen Wahrscheinlichkeitswerten des Originaltextes und der Perturbationen, kann nun eine binäre Klassifizierung vorgenommen werden, ob der Text von einem Modell generiert wurde oder nicht.

3.4 Kompressionsverfahren

Da von Sprachmodellen generierte Texte oft wiederkehrende Muster aufweisen, bietet die Anwendung von Kompressionsalgorithmen, die wiederholende Sequenzen erkennen, eine Möglichkeit zur Textklassifizierung. Torrey (2023) stellt ein entsprechendes Verfahren vor.

Zu Beginn wird eine Sammlung von Texten, die von einem beliebigen Sprachmodell generiert wurden, mithilfe eines gängigen Kompressionsalgorithmus komprimiert. Hierbei wird das Verhältnis zwischen der Größe des komprimierten Textkorpus und der Größe des unkomprimierten Textkorpus berechnet. Nun können einzelne Texte überprüft werden. Dafür wird der Textsammlung der zu überprüfende Text angefügt und erneut komprimiert. Falls die Bestandteile des Textes bereits in der Nachschlagetabelle des Kompressionsalgorithmus vollständig abgebildet sind, würde das Verhältnis vom komprimiertem Textkorpus zu dem unkomprimiertem

Textkorpus abnehmen. Dies weist auf einen von KI-generierten Text hin. Wenn das Verhältnis hingegen größer wird, deutet dies auf eine Abweichung in der Textstruktur hin, was gegen eine KI-Generierung spricht.

Um dieses Verfahren zu implementieren, wäre es demnach zuerst notwendig, Texte von einem Sprachmodell, auf das geprüft werden soll, generieren zu lassen. Diese Texte bilden nun die Vergleichsgrundlage und werden mit einem gängigen Kompressionsalgorithmus, welcher die Größe einer Datensequenz reduziert, indem Muster in den Daten erkannt und zusammengefasst werden, komprimiert. Danach wird die Größe aller komprimierten Texte zu der Größe aller unkomprimierter Texte in ein Verhältnis gesetzt bzw. die Größe der komprimierten Texte durch die Größe der unkomprimierten Texte dividiert. Wenn nun ein LLM-generierter Text überprüft wird, wird er dem Textkorpus angefügt und dieser erneut komprimiert. Da Sprachmodelle Muster in der Textgenerierung aufweisen, sollten für den neu angefügten Text Zusammenfassungen des Kompressionsalgorithmus vorliegen. Wenn man nun das Verhältnis von komprimierter Größe zu unkomprimierter Größe erneut berechnet, fällt auf, dass erstere weniger stark als letztere angestiegen ist, der Bruch also kleiner wird. Dies deutet auf eine LLM-Generierung hin. Wenn dieses Verfahren analog für einen nicht KI-generierten Text durchgeführt wird, findet der Algorithmus weniger Möglichkeiten den Text mit vorliegenden Mustern zusammenzufassen, die Größe der komprimierten Texte steigt also ähnlich stark wie die Größe der unkomprimierten Texte. Dabei wird der Bruch größer, was gegen eine KI-Generierung spricht.

Der Erfolg dieses Verfahrens hängt maßgeblich von der anfänglich gewählten Textsammlung ab. Da die Implementierung im Vergleich zu anderen Verfahren einfach und die Ausführungszeit kurz ist, bietet es sich an, die Textsammlung auf den jeweiligen Anwendungsfall abzustimmen.

3.5 Kennzeichnungsverfahren

Kennzeichnungsverfahren (engl. White Box Methoden) bieten die Möglichkeit, im LLM-generierten Text eine Art Wasserzeichen (engl. Watermarking) einzufügen, welches nur für den Ersteller des Sprachmodells ersichtlich und im Output nicht zu erkennen ist (Kirchenbauer et al. 2023). Zum jetzigen Zeitpunkt gibt es noch keine relevanten Sprachmodelle, welche Kennzeichnungsverfahren verwenden. Dies ist unter anderem darin begründet, dass bisher noch keine Wege gefunden worden sind, alle Umgehungsmöglichkeiten des Watermarkings zu verhindern.

Ein einfacher Ansatz für Watermarking ist es, basierend auf dem Hash des vorherigen Tokens, das Vokabular des LLMs in eine Liste mit erlaubten und nicht erlaubten Tokens einzuteilen, welche jeweils 50 % des Gesamtvokabulars beinhalten. Dies ermöglicht eine verlässliche Klassifikation, denn die Wahrscheinlichkeit, dass ein menschlich generierter Text nur Tokens aus den erlaubten Listen enthält, liegt bei einer Sequenz von N Tokens bei $1/2^N$, bei maschinengeneriertem Text immer bei 1.

Dieser Ansatz hat eine erhebliche Schwachstelle. In manchen Fällen gibt es Tokens, auf die nur ein bestimmter Token sinnvoll folgen kann. So würde z. B. ein Text über bekannte Wissenschaftler inhaltlich unbrauchbar werden, wenn als Fol-

getoken für „Albert“, „Einstein“ in der Liste, der nicht erlaubten Tokens stehen würde. Dieses Problem kann durch ein Kennzeichnungsverfahren gelöst werden, welches inhaltlich wichtige Tokens nicht ausschließt. Um das zu erreichen, wird die Wahrscheinlichkeitsverteilung der Sprachmodelle genutzt. In Fällen, bei denen einer bis wenige Tokens eine sehr hohe Wahrscheinlichkeit besitzen, werden diese zur erlaubten Liste hinzugefügt. In Fällen, bei denen viele Tokens eine moderate Wahrscheinlichkeit besitzen, erfolgt die Zuordnung zufällig. Dieses Verfahren bietet den Vorteil, dass es den generierten Text inhaltlich nicht einschränkt, aber dennoch eine starke Kennzeichnung ermöglicht. Um die Kennzeichnungsstärke zu optimieren, wird eine Listenlänge erlaubter Tokens von ca. 10 % des Gesamtvokabulars empfohlen.

Es gibt einige Möglichkeiten Kennzeichnungsverfahren auch mit geringem Aufwand zu umgehen, die nur teilweise unterbunden werden können. Der offensichtlichste Weg das Wasserzeichen zu entfernen, ist die Paraphrasierung von Texten. Um das Watermarking signifikant unkenntlich zu machen, müssten aber ca. ein Viertel der generierten Tokens durch unerlaubte ersetzt werden.

Dabei werden einige Wörter durch andere ersetzt, was die Listen der erlaubten und nicht erlaubten Tokens verändert. Die Paraphrasierung kann durch einen Menschen erfolgen, was zeitintensiv ist oder zeiteffizient durch ein schwächeres Sprachmodell, worunter die inhaltliche Qualität leiden könnte. Von der Paraphrasierung ausgenommene Abschnitte können außerdem ausreichen um ein statistisch relevantes Erkennungssignal zu liefern (Kirchenbauer et al. 2023).

Die effizienteste Umgehungsmethode sind sogenannte „Emoji-Attacks“ (Goodside 2023). Dabei wird ein LLM per Prompteingabe aufgefordert einen Text zu verfassen und dabei zwischen jedem Wort ein Emoji einzufügen. Diese Emojis lassen sich automatisiert wieder aus dem Text entfernen, was die Listen mit den erlaubten und nicht erlaubten Tokens verändert und das Wasserzeichen unbrauchbar macht. Dabei ist es nicht notwendig, mit Emojis zu arbeiten, denn jede Anweisung an das LLM seinen Output zu verändern, erzielt denselben Effekt. Denkbar wäre hierbei, das LLM zu trainieren solche Anweisungen nicht zu erfüllen.

Andere Angriffe auf die Kennzeichnung beinhalten z. B. eine Veränderung von Zeichenkodierungen, welche sich durch Input-Normalisierungen mitigieren lassen. Auch wenn die verschiedenen Umgehungsverfahren nicht vollständig verhindert werden können, ist es doch möglich, eine nicht erkennbare Generierung von Texten wesentlich zu erschweren. Dies könnte in vielen Situationen dazu führen, dass Personen, in Anbetracht des erheblichen Aufwands, davon absehen Texte zu generieren, was ein Teilerfolg im Kampf gegen ungekennzeichnete Verwendung von LLMs wäre (Kirchenbauer et al. 2023).

4 Vergleich und Diskussion der Ansätze

Die nachfolgende Tab. 1 stellt die verschiedenen Klassifizierungs- und Kennzeichnungsverfahren übersichtlich dar. Diese Übersicht ist das Ergebnis einer strukturierten Literaturanalyse. Neben den jeweiligen Autoren werden die zugrundeliegenden Verfahren und Methoden zugeordnet. Weiterhin stellt die Tabelle für jeden Ansatz

Tab. 1 Ansätze zur Erkennung von LLM-generierten Texten

Autoren	Verfahren	Methode	LLM	Datensatz
Chakraborty et al. (2023b)	Statistisch, Zero Shot	Supervised	GPT-2	XSum Wikipedia from Squad IMDb Reviews Kaggle Fake News
Deng et al. (2023)	Statistisch, Deep Learning	Gaussian Process	GPT-2, LLaMA 65B	XSum Wikipedia from SQuAD Subreddit/r/ WritingPrompts
Fröhling und Zubiaga (2021)	Statistisch	Logistic Regression SVM Neural Network Random Forest	GPT-2 GPT-3 Grover	Grover realNews
Gallé et al. (2021)	Deep Learning	BERT	GPT-2	Gutenberg
Guo et al. (2023)	Statistisch, Deep Learning	Logistic Regression, RoBERTa	GPT-3.5	HC3
Ippolito et al. (2020)	Statistisch, Deep Learning	Logistic Regression, BERT	GPT-2	Web Text
Mitchell et al. (2023)	Zero Shot	Log Probability	GPT-2 OPT2.7 Neo2.7 GPT-J NeoX	XSum Wikipedia from SQuAD PubMedQA Subreddit/r/ WritingPrompts
Rodriguez et al. (2022)	Deep Learning	RoBERTa	GPT-2	Semantic Scholar, CORE
Torrey (2023)	Kompression	LZMA	ChatGPT, Bard Bing	GPT-2/-3 Datensätze
Verma et al. (2023)	Statistisch, Deep Learning	Unigramme Trigramme Logistic Regression zwei GPT-3 Modelle	GPT-3.5	Subreddit/r/ WritingPrompts Reuters 50-50 BAWE Corpus

dar, welches Sprachmodell mit dem Ansatz erkannt werden soll. Darüber hinaus werden die genutzten Datensätze benannt.

Bemerkenswert ist, dass sich die bisherigen Ansätze trotz ihrer zeitlichen Aktualität auf die Erkennung von älteren GPT-Versionen konzentrieren und die Entwicklung von GPT-4 nicht berücksichtigen. Gerade GPT-4 verfügt über eine bedeutend größere Anzahl von Parametern und wurde mit einem viel größeren Datensatz trainiert, was die Vielfältigkeit und Präzision der generierten Texte erhöhen und die Erkennung sicherlich erschweren wird.

Tab. 2 Übersicht von implementierten Detektoren, Stand: Oktober 2023 in Anlehnung an Weber-Wulff et al. (2023)

Name	Website	Verfahren
Compilatio AI Detector	https://ai-detector.compilatio.net/	Deep Learning
Content at Scale AI Detector	https://contentatscale.ai/ai-content-detector/	Deep Learning, statistisch
Crossplag	https://crossplag.com/ai-content-detector/	Deep Learning, statistisch
Winston AI	https://gowinston.ai/	Deep Learning, statistisch
GPTZero	https://gptzero.me/	Deep Learning, statistisch
GPT-2 Output Detector	https://openai-openai-detector.hf.space/	Deep Learning
Plagiarism Check	https://plagiarismcheck.org/	Unbekannt
Writefull GPT Detector	https://x.writefull.com/gpt-detector	Unbekannt
Writer AI Content Detector	https://writer.com/ai-content-detector/	Unbekannt
ZeroGPT	https://www.zerogpt.com/	Deep Learning

Die verschiedenen Beiträge benutzen unterschiedliche Trainings- und Testdatensätze, was einen Vergleich erschwert. Es fällt auf, dass die Ansätze hohe Erkennungsraten für die gewählten Testdaten aufweisen, die bei der Benutzung eines anderen Testdatensatzes stark einbrechen. Mitchell et al. (2023) nennen für ihren Ansatz DetectGPT Erkennungsraten von mindestens 93 %, wohingegen Verma et al. (2023) in ihrem Vergleich unterschiedlicher Erkennungsansätze auf Erkennungsraten von maximal 78 % für DetectGPT kommen. Das lässt sich durch die unterschiedlichen Testdatensätze erklären und zeigt auf, dass die skizzierten Erkennungsraten nur eine bedingte Aussagekraft haben. Für das Benchmarking von Detektoren sollten unterschiedliche und vielfältige Testdatensätze verwendet werden, um die Übertragbarkeit der Ergebnisse auf realitätsnahe Szenarien besser gewährleisten zu können.

Die verschiedenen Ansätze zur Erkennung von LLM-generierten Texten fließen in unterschiedliche Implementierungen und Detektoren ein. Tab. 2 stellt diese übersichtsartig vor, welche im Rahmen einer Literaturanalyse identifiziert worden sind.

Einige Detektoren, wie CheckforAi (https://checkforai.com) oder der Detektor von OpenAI, sind bereits wieder offline. Bei einigen Detektoren wird der dahinterstehende Ansatz nicht oder nur sehr oberflächlich beschrieben. Nicht jeder Detektor ist für die Erkennung von jedem Sprachmodell geeignet. Es gibt Detektoren, die nur für die Erkennung von GPT-2, wie der GPT-2 Output Detector, konzipiert sind, ein Sprachmodell, das als veraltet angesehen werden kann, da es mit GPT-3.5 und GPT-4 erhebliche Weiterentwicklungen erfahren hat. Ein wichtiger Unterschied besteht auch in der minimal notwendigen Anzahl der Wörter des zu testenden Textes und in Bezug auf die Sprachen, in denen eine Erkennung durchgeführt werden kann. Englische Texte zu überprüfen ist oftmals möglich, Texte in deutscher Sprache eher selten, was im Zusammenhang mit der größeren Verfügbarkeit englischer Trainingsdaten steht.

5 Überlegungen zu Trainingsdatensätzen

Für das Training der genannten Detektoren wurden unterschiedliche Datensätze verwendet, die einen Einfluss auf die Güte dieser Detektoren für die jeweilige Sprache haben, weshalb die Wahl oder Konstruktion des Datensatzes von entscheidender Bedeutung ist. Keiner der Datensätze ist auf Deutsch. Eine Möglichkeit, um einen deutschen Trainingsdatensatz zu erhalten, ist die Übersetzung der Datensätze ins Deutsche. Die verschiedenen Ansätze basieren auf den folgenden Datensätzen.

In dem Datensatz Subreddit/r/WritingPrompt, der von Menschen geschriebene Texte beinhaltet und in drei Ansätzen verwendet wird (Tab. 1), kann jeder registrierte User auf einen vorgegebenen Prompt mit einer Antwort reagieren. In der Regel handelt es sich dabei um Kurzgeschichten oder kreative Texte in englischer Sprache. Im Projekt British Academic Written English Corpus (BAWE) wurden von Studenten geschriebene Aufsätze aus verschiedenen Fachgebieten archiviert. Dieser Datensatz ist nur in englischer Sprache verfügbar. Einen ähnlichen Ansatz bietet der CORE-Datensatz. Alle öffentlich verfügbaren wissenschaftlichen Publikationen in englischer Sprache können mit dieser API abgerufen werden. Grover ist ein Framework, um Fake News zu erkennen. Der Datensatz Grover realNews besteht aus Nachrichtenartikeln aus dem Internet. Es ist mit Grover zusätzlich möglich Fake News Artikel zu generieren. Der Gutenberg Datensatz beinhaltet Bücher und Werke von Autoren, deren Urheberrecht erloschen ist und deren Texte frei verfügbar sind. Das Angebot umfasst viele Sprachen, der Großteil des Datensatzes ist in englischer Sprache. HC3 (Human ChatGPT Comparison Corpus) beinhaltet die Antworten von Menschen sowie von ChatGPT auf dieselbe Frage auf Englisch und Chinesisch. Der Datensatz IMDb Review beinhaltet von Usern geschriebene Rezensionen zu Filmen in englischer Sprache. Kaggle Fake News ist ein Datensatz, der für einen Wettbewerb auf Kaggle bereitgestellt wurde, um Modelle zur Fake News Identifizierung zu entwickeln. Es handelt sich um echte Nachrichtenartikel sowie Fake News Artikel, jeweils in englischer Sprache. Im PubMedQA Datensatz wurden medizinische Fragen von Experten nur mit den Optionen ja/nein/vielleicht beantwortet. Es handelt sich um einen englischsprachigen Datensatz. Aus dem Datensatz RCV1 ist der Reuters-50-50 entstanden. Das Corpus besteht aus englischen Texten von genau 50 Autoren. Semantic Scholar bietet eine Sammlung öffentlich verfügbarer wissenschaftlicher Publikationen in englischer Sprache über den Zeitraum von 1985 bis 2017, wobei die Daten über eine API bereitgestellt werden. SQuAD (Stanford Question Answering Dataset) ist eine Sammlung von Frage-Antwort-Paaren, die sich auf Wikipedia Artikel beziehen. Die von Menschen beantworteten Fragen beziehen sich auf die Artikel. Fragen und Antworten sind in englischer Sprache. Der letzte Datensatz XSum (Extreme Summarization) ist eine Sammlung von BBC-Nachrichtenartikeln und deren Kurzzusammenfassungen von Menschen auf Englisch. Tab. 3 stellt die Quellen und Sprachen der Datensätze dar.

Für das Trainieren von Modellen sollte der Datensatz möglichst groß sein. Grundsätzlich muss darauf geachtet werden, dass die Verteilung der einzelnen Klassen wirklichkeitsnah ist. Bei der Erstellung des Trainingsdatensatzes sind unterschiedliche Vorgehen denkbar. Ein Verfahren ist das manuelle Labeln, das naturgemäß kostenintensiv und nur begrenzt skalierbar ist, wenngleich es eine hohe Qualität

Tab. 3 Datensätze für das Training von existierenden Detektoren

Datensatz	Sprache	Website
/r/ WritingPrompts	En	https://www.reddit.com/r/WritingPrompts/
BAWE	En	https://www.coventry.ac.uk/research/research-directories/current-projects/2015/british-academic-written-english-corpus-bawe/
CORE	En	https://core.ac.uk/services/dataset
Grover	En	https://github.com/rowanz/grover
Gutenberg	En	https://www.gutenberg.org/
HC3	En, cn	https://github.com/Hello-SimpleAI/chatgpt-comparison-detection
IMDb Review	En	https://www.imdb.com/
Kaggle Fake News	En	https://www.kaggle.com/competitions/fake-news/data
PubMedQA	En	https://pubmedqa.github.io/
Reuter-50-50	En	https://archive.ics.uci.edu/dataset/217/reuter +50+50
Semantic Scholar	En	https://www.semanticscholar.org/
SQuAD	En	https://rajpurkar.github.io/SQuAD-explorer/
XSum	En	https://github.com/EdinburghNLP/XSum/tree/master/XSum-Dataset

verspricht. Zur vorliegenden Fragestellung ist eine große Menge an Trainingsdaten notwendig. Diese Daten sind in einer großen Menge im Internet verfügbar (Bewertungstexte, Blogbeiträge oder Nachrichtenartikel) und durch eine Eingrenzung des Zeitraums vor 2020, kann mit relativ großer Sicherheit davon ausgegangen werden, dass es sich um von Menschen geschriebene Texte handelt.

Bestehen die Daten nur aus Texten, ist es erforderlich die Zusammensetzung genau zu analysieren. So muss zumindest in der Europäischen Union der Datenschutz angemessen berücksichtigt werden. In dem Zuge müssen personenbezogene Daten und Informationen bei der Trainingsdatensatzerstellung entfernt werden, wie z. B. Namen oder Personen.

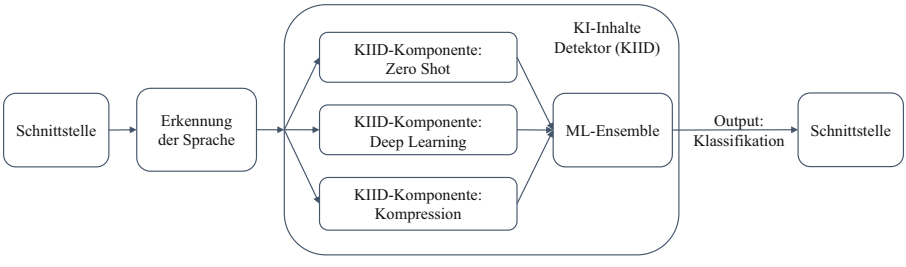

Abb. 1 Architektur des KI-Inhalte Detektors (KIID)

6 Erkennung deutscher KI-generierter Texte

Die vorgestellten Beiträge konzentrieren sich bei der Erkennung von KI-generierten Inhalten auf englischsprachige Texte. Für deutschsprachige Texte besteht bisher noch eine Forschungslücke, die der nachfolgende Ansatz adressiert. Dieser Vorschlag der Autoren vereint verschiedene bereits existierende Ansätze, die in den vorherigen Kapiteln besprochen worden sind.

Die Abb. 1 stellt die Architektur des KI-Inhalte Detektor (KIID) dar. Dieser Detektor besteht aus fünf zentralen Komponenten. Die erste Komponente ist eine Sprachenerkennung, die prüft, ob es sich tatsächlich um einen deutschsprachigen Text handelt. Der eigentliche Detektor besteht aus vier Komponenten und führt drei Verfahren zur Erkennung von KI-generierten Texten zusammen.

Es wird ein Zero Shot, ein Deep Learning und ein Kompressionsverfahren auf den deutschsprachigen Text angewandt. Das Deep Learning System basiert auf einem BERT-Modell mit Finetuning. Für das Deep Learning und das Kompressionsverfahren werden gelabelte Trainingsdaten gebraucht. Die von Menschen geschriebenen Texte werden auf Basis des FANG-COVID-Datensatzes (Mattern et al. 2021) erstellt, der Nachrichtenartikel unterschiedlicher deutscher Zeitungen beinhaltet. Diesem Datensatz werden weitere News Beiträge (ca. 460k) und ca. 500k Produktbewertungen von Amazon, die jeweils vor dem Jahr 2020 abgegeben worden sind, hinzugefügt. Die KI-generierten Texte werden mit Hilfe von VICUNA-13B und GPT-4 (jeweils 500k Trainingsdaten) erzeugt. Diese Texte werden per entsprechender Prompteingabe derart generiert, dass sie sowohl News Beiträge als auch Produktbewertungen behandeln. Diese bilden das Gegengewicht zu den durch Menschen verfassten Texten.

Die Ergebnisse der drei KIID-Komponenten fließen in ein ML-Ensemble ein, indem die Teilergebnisse dynamisch gewichtet werden, um das finale binäre Ergebnis zu erhalten. Die Nutzung der drei unterschiedlichen Verfahren wird zu einer besseren Erkennungsrate für verschiedenartige Texte führen und verspricht eine höhere Generalisierbarkeit vom KIID.

7 Zusammenfassung, Limitationen und Ausblick

Im vorliegenden Beitrag wurde die Entwicklung der Sprachmodelle hin zu den großen Sprachmodellen präsentiert. Durch die fortschreitende Entwicklung der Sprachmodelle ergeben sich neben den Möglichkeiten auch vielfältige Probleme, weshalb eine Erkennung von LLM-generierten Texten notwendig ist. Dieser Beitrag stellt die unterschiedlichen Ansätze bekannter Erkennungsverfahren dar: statistische, Deep-Learning-basierte, Zero-Shot und kompressionsorientierte Verfahren. Daneben werden Kennzeichnungsverfahren präsentiert. Nach dem tabellarischen Vergleich der in der Literatur vorgestellten Verfahren werden implementierte Softwaredetektoren präsentiert. Dieser Vergleich von existierenden Detektoren stellt nur eine Momentaufnahme dar, die nicht vollständig ist, aber einen ersten Überblick in das Thema gibt. Eine stetige Weiterentwicklung in diesem Bereich ist zu erwarten. Gerade bei den Softwaredetektoren ist ersichtlich, dass einige Ansätze bereits nicht mehr ver-

fügbar oder nicht die Fähigkeit haben, KI-generierte Texte von den neusten LLMs, wie GPT-4, zu erkennen. Im Anschluss werden Überlegungen zum Entwurf eines Trainingsdatensatzes aufgezeigt, wodurch die Grundlage für einen eigenen Ansatz zur Erkennung von KI-generierten Texten in deutscher Sprache geschaffen wird. Darüber hinaus wird die Architektur und das Design des eigenen Ansatzes, dem KIID, vorgestellt und beschrieben. Bisher sind noch keine Daten zu der Erkennungsquote verfügbar, da der KIID aktuell implementiert wird. Dieser vorgestellte und eingeführte Ansatz wird in einem folgenden Beitrag weiterführend beschrieben. Dabei stehen vor allem die Ergebnisse und die Leistungsfähigkeit, wie Erkennungsquote und Robustheit, im Vordergrund. Aufbauend auf dem Beitrag ist bereits geplant, dass die Evaluierung mit verschiedenen Testdatensätzen von unterschiedlichen Sprachmodellen durchgeführt wird. Weiterhin wird auch der Vergleich zu den bestehenden Ansätzen gezogen. In einem weiteren Beitrag werden die anderen verfügbaren Detektoren mit verschiedenen Datensätzen evaluiert und untereinander verglichen, um tatsächliche Erkennungsraten der Detektoren zu bestimmen.

Funding Open Access funding enabled and organized by Projekt DEAL.

Literatur

Alzubaidi L, Zhang J, Humaidi AJ, Al-Dujaili A, Duan Y, Al-Shamma O, Santamaría J, Fadhel MA, Al-Amidie M, Farhan L (2021) Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8:53. https://doi.org/10.1186/s40537-021-00444-8

Chakraborty M, Tonmoy SMTI, Zaman SMM, Sharma K, Barman NR, Gupta C, Gautam S, Kumar T, Jain V, Chadha A, Sheth AP, Das A (2023a) Counter Turing test CT^2: aI-generated text detection is not as easy as you may think—introducing AI detectability index. https://arxiv.org/pdf/2310.05030.pdf

Chakraborty S, Bedi AS, Zhu S, An B, Manocha D, Huang F (2023b) On the possibilities of AI-generated text detection. https://arxiv.org/pdf/2304.04736.pdf

Chan B, Schweter S, Möller T (2020) German's next language model. https://arxiv.org/pdf/2010.10906.pdf

Chen S, Beeferman D, Rosenfeld R (1998) Evaluation metrics for language models. Carnegie Mellon University

Deng Z, Gao H, Miao Y, Zhang H (2023) Efficient detection of LLM-generated texts with a Bayesian surrogate model. https://arxiv.org/pdf/2305.16617.pdf

Devlin J, Chang M-W, Lee K, Toutanova K (2018) BERT: pre-training of deep bidirectional transformers for language understanding. https://arxiv.org/pdf/1810.04805.pdf

Fröhling L, Zubiaga A (2021) Feature-based detection of automated language models: tackling GPT-2, GPT-3 and Grover. PeerJ Comput Sci 7:e443. https://doi.org/10.7717/peerj-cs.443

Gallé M, Rozen J, Kruszewski G, Elsahar H (2021) Unsupervised and distributional detection of machine-generated text. https://arxiv.org/pdf/2111.02878.pdf
Goodside R (2023) There are adversarial attacks for that proposal as well—in particular, generating with emojis after words and then removing them before submitting defeats it. https://twitter.com/goodside/status/1610682909647671306
Gordin MD (2016) The Dostoevsky machine in Georgetown: scientific translation in the Cold War. Ann Sci 73:208–223. https://doi.org/10.1080/00033790.2014.917437
Guo B, Zhang X, Wang Z, Jiang M, Nie J, Ding Y, Yue J, Wu Y (2023) How close is ChatGPT to human experts? Comparison corpus, evaluation, and detection. https://arxiv.org/pdf/2301.07597.pdf
Harguess J, Ward CM (2022) Is the next winter coming for AI? Elements of making secure and robust AI. In: 2022 IEEE Applied Imagery Pattern Recognition Workshop (AIPR). 11–13 Oct. 2022. IEEE, Piscataway, S 1–7
Haupt CE, Marks M (2023) AI-generated medical advice-GPT and beyond. JAMA 329:1349–1350. https://doi.org/10.1001/jama.2023.5321
Hazell J (2023) Large language models can be used to effectively scale spear phishing campaigns. https://arxiv.org/pdf/2305.06972.pdf
Ippolito D, Duckworth D, Callison-Burch C, Eck D (2020) Automatic detection of generated text is easiest when humans are fooled. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics:, S 1808–1822 https://doi.org/10.18653/v1/2020.acl-main.164
Kirchenbauer J, Geiping J, Wen Y, Katz J, Miers I, Goldstein T (2023) A watermark for large language models. https://arxiv.org/pdf/2301.10226.pdf
Kirchner JH, Ahmad L, Aaronson S, Leike J (2023) New AI classifier for indicating AI-written text. https://openai.com/blog/new-ai-classifier-for-indicating-ai-written-text
Liu Y, Ott M, Goyal N, Du Jingfei, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L, Stoyanov V (2019) RoBERTa: a robustly optimized BERT pretraining approach. https://arxiv.org/pdf/1907.11692.pdf
Manning CD, Schütze H (2005) Foundations of statistical natural language processing. MIT Press, Cambridge
Mattern J, Qiao Y, Kerz E, Wiechmann D, Strohmaier M (2021) FANG-COVID: a new large-scale benchmark dataset for fake news detection in German. Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER), S 78–91 https://doi.org/10.18653/v1/2021.fever-1.9
Mitchell E, Lee Y, Khazatsky A, Manning CD, Finn C (2023) DetectGPT: zero-shot machine-generated text detection using probability curvature. https://arxiv.org/pdf/2301.11305.pdf
Orenstrakh MS, Karnalim O, Suarez CA, Liut M (2023) Detecting LLM-generated text in computing education: a comparative study for ChatGPT cases. https://arxiv.org/pdf/2307.07411.pdf
Peng L, Zhang Y, Shang J (2023) Generating efficient training data via LLM-based attribute manipulation. https://arxiv.org/pdf/2307.07099.pdf
Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, Zhou Y, Li W, Liu PJ (2019) Exploring the limits of transfer learning with a unified text-to-text transformer. https://arxiv.org/pdf/1910.10683.pdf
Rodriguez J, Hay T, Gros D, Shamsi Z, Srinivasan R (2022) Cross-domain detection of GPT-2-generated technical text. In: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, S 1213–1233 https://doi.org/10.18653/v1/2022.naacl-main.88
Rosenblatt F (1958) The perceptron: a probabilistic model for information storage and organization in the brain. Psychol Rev 65:386–408. https://doi.org/10.1037/h0042519
Sarker IH (2022) AI-based modeling: techniques, applications and research issues towards automation, intelligent and smart systems. SN Comput Sci 3:158. https://doi.org/10.1007/s42979-022-01043-x
Shuai Z, Xiaolin D, Jing Y, Yanni H, Meng C, Yuxin W, Wei Z (2022) Comparison of different feature extraction methods for applicable automated ICD coding. BMC Med Inform Decis Mak 22:11. https://doi.org/10.1186/s12911-022-01753-5
Su J, Zhuo TY, Mansurov J, Wang D, Nakov P (2023) Fake news detectors are biased against texts generated by large language models. https://arxiv.org/pdf/2309.08674.pdf
Sundermeyer M, Schlüter R, Ney H (2012) LSTM neural networks for language modeling. Interspeech
Tang R, Chuang Y-N, Hu X (2023) The science of detecting LLM-generated texts. https://arxiv.org/pdf/2303.07205.pdf
Tappert CC (2019) Who is the father of deep learning? In: 2019 International Conference on Computational Science and Computational Intelligence (CSCI)
Torrey J (2023) Meet „ZipPy“, a fast AI LLM text detector. https://blog.thinkst.com/2023/06/meet-zippy-a-fast-ai-llm-text-detector.html

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. https://arxiv.org/pdf/1706.03762.pdf
Venkit PN, Gautam S, Panchanadikar R, Huang T-H, Wilson S (2023) Nationality bias in text generation. https://arxiv.org/pdf/2302.02463.pdf
Verma V, Fleisig E, Tomlin N, Klein D (2023) Ghostbuster: detecting text ghostwritten by large language models. https://arxiv.org/pdf/2305.15047.pdf
Vicuna (2023) Vicuna: an open-source chatbot impressing gpt-4 with 90 %* Chatgpt quality. https://vicuna.lmsys.org/
Weber-Wulff D, Anohina-Naumeca A, Bjelobaba S, Foltýnek T, Guerrero-Dib J, Popoola O, Šigut P, Waddington L (2023) Testing of detection tools for AI-generated text. https://arxiv.org/pdf/2306.15666.pdf
Weizenbaum J (1966) ELIZA—a computer program for the study of natural language communication between man and machine. Communications of the ACM
Yao Y, Duan J, Xu K, Cai Y, Sun E, Zhang Y (2023) A survey on Large Language Model (LLM) security and privacy: the good, the bad, and the ugly. http://arxiv.org/pdf/2312.02003.pdf