technische universität
dortmund

Bachelorarbeit

Using large language models for anomaly detection on text datasets

Anton Häusler
Juni 2023

Gutachter:
Prof. Dr. Emmanuel Müller
Simon Klüttermann

Technische Universität Dortmund
Fakultät für Informatik
Chair of Data Science and Data Engineering
(LS IX)
https://ls9-www.cs.tu-dortmund.de/

Inhaltsverzeichnis

1 Einführung

Die wachsende Bedeutung in der Analyse von Textdaten macht es notwendig, Verfahren zu entwickeln, die keine weitere Unterstützung von Experten benötigen, sondern komplett automatisiert erfolgen. Der Quelltext von Websiten, digitalisierte Dokumente oder auch Texte in sozialen Netzwerken - die Digitalisierung hat in der Vergangenheit und wird in der Zukunft komplexere und zunehmend mehr Textdaten generieren. Mit der steigenden Rechenleistung (Moores Law) ist es möglich, Modelle mit Billionen von Parametern zu entwickeln, die in der Lage sind, unsere Sprache, ein komplexes Konstrukt, nachzubilden.
Die Analyse von großen Datenmengen eröffnet neue Wettbewerbschancen für Unternehmen und kann gleichzeitig potentielle Risiken minimieren.

Die Anomalieerkennung ist eine numerische Aufgabe. Für eine Anwendung auf Textdaten müssen diese in einem ersten Schritt in eine Vektorrepräsentation transformiert werden.
In früheren Arbeiten wurden geeignete Features erstellt und ein Text durch die Ausprägung in den einzelnen Features repräsentiert. Durch die Integration dieser Features konnte eine syntaktische Analyse von Texten durchgeführt werden. Die Erstellung von geeigneten Features für eine semantische Beschreibung stellt jedoch weiterhin eine Herausforderung dar. Im Bereich des natural language processing (NLP) gab es in den letzten Jahren vielversprechende Fortschritte. Wendepunkt der Entwicklung war die Einführung der Transformer Architektur [33]. Anders als bei den recurrent neural networks (RNNs), die sequentielle Eingaben benötigen, ermöglicht diese Architektur eine parallele Eingabe von Daten. Der daraus resultierende Geschwindigkeitsvorteil macht es möglich, größere und tiefere Netzwerke mit mehr Daten zu trainieren, um bessere Ergebnisse in NLP Aufgaben zu erzielen.
Die sogenannten large language model (LLMs) transformieren den Eingabetext in eine interne Repräsentation, die die Semantik des Textes darstellt.
Im Rahmen dieser Arbeit wird die Leistungsfähigkeit verschiedener Kombinationen von LLMs und Anomalieerkennungsalgorithmen untersucht. Das Ziel ist die Entwicklung einer Basisarchitektur auf die, durch weitere Modifikation, aufgebaut werden kann.

Die Arbeit ist wie folgt aufgebaut. Zu Beginn werden im Kapitel 2 die theoretischen Grundlagen der Arbeit erläutert. Kapitel 3 gibt ein Überblick über verwandte Arbeiten. Der generelle Aufbau der Architektur wird in Kapitel 5 dargestellt. Im anschließenden Kapitel 6 werden die Experimente beschrieben und die Ergebnisse bewertet. Die aus den Ergebnissen abgeleitete Basisarchitektur und ein Ausblick auf zukünftige Arbeiten werden in Kapitel 7 vorgestellt.

2 Theoretische Grundlage

In diesem Kapitel werden alle Grundlagen erläutert, auf denen die Arbeit aufbaut und die für das Verständnis relevant sind.

2.1 Natural Language

Es wird zwischen natürlichen und formalen Sprachen unterschieden. Die menschliche Sprache gehört zu den natürlichen Sprachen. Ein Beispiel für eine formale Sprache ist die Programmiersprache C. Die natürlichen Sprachen wurden nicht spezifisch für ein Anwendungsgebiet definiert, sondern haben sich über die Zeit natürlich entwickelt und dienen der Kommunikation. Die natürlichen Sprachen sind oft nicht eindeutig und nur durch den semantischen Kontext zu verstehen. Für den Menschen ist die Verwendung von doppeldeutigen Wörtern einfacher, als sich für jede Bedeutung ein eigenes Wort zu merken.

Im Unterschied dazu, ist es für Computer eine nicht triviale Aufgabe die Bedeutung zu erkennen. Die unterschiedlichen Sprachen, Dialekte, Redewendungen aber auch Rechtschreib- und Grammatikfehler erschweren dem Computer den Analyseprozess der menschlichen Sprache. Im nächsten Kapitel 2.2 wird der Bereich eingeleitet, der sich mit dem Problem der Analyse von natürlicher Sprache beschäftigt.

2.2 Natural Language Processing

Bei dem natural language processing (NLP) handelt es sich um ein interdisziplinäres Forschungsgebiet, das sich mit der Verarbeitung der menschlichen Sprache durch Computer befasst. NLP umfasst Techniken aus Bereichen der Informatik, Linguistik und des maschinellen Lernens, um die schriftliche oder gesprochene menschliche Sprache zu analysieren, zu verstehen und zu generieren.

2.2.1 Anwendungen

Im Bereich des NLP gibt es viele verschiedene Anwendungen. Dazu zählen:

- *Translation* Die Aufgabe besteht darin, einen gegeben Textabschnitt in eine andere, vorgegebene Sprache zu übersetzen.

- *Text Summarization* Die Aufgabe besteht darin, einen gegeben Text inhaltlich zusammenzufassen.

- *Question Answering* Die Aufgabe besteht darin, eine Frage korrekt zu beantworten. Unter Umständen ist neben der Frage auch ein Text gegeben, dann soll die Antwort Bezug zum Text nehmen.

Die oben beschriebenen Aufgaben verlangen eine Generierung von Sätzen. Im Anwendungsfall der Outlier Detection handelt es sich um überwachende Aufgaben, in dem der Algorithmus keine Sätze auf Basis der Eingaben generiert, sondern Abweichungen in den Eingaben erkennt und gegebenenfalls alarmiert. Mögliche Beispiele sind:

- Die Nachrichten eines Systems überwachen. Bei ungewohnten Nachrichten ein Hinweis ausgeben.
- Den Quelltext von neu erstellten Websiten überwachen. Bei potentieller Malware eine Meldung ausgeben.

Neben diesen genannten Beispielen gibt es eine Vielzahl an möglichen Anwendungen der Anomaly Detection auf Texten.

2.2.2 Tokenization

Im Bereich des NLP gibt es zwei große Kernaufgaben. Die erste ist, einen zu analysierenden Text in eine Vektorrepräsentation umzuwandeln, die den Text sowohl syntaktisch, als auch semantisch widerspiegelt. Die zweite Aufgabe ist dann, aus dieser Vektorrepräsentation die gewünschte Ausgabe zu generieren.
Die Zusammenfassung der gesamten Bedeutung eines Textes in einen einzigen Vektor stellt eine Herausforderung dar. Eine Herangehensweise besteht darin, den Text zunächst in kleinere Abschnitte zu unterteilen und für jeden Abschnitt einen Vektor zu generieren und diese dann anschließend zu einem geeigneten Textvektor zu kombinieren. Diese Unterteilung wird in einem Preprocessing Schritt namens Tokenization durchgeführt.
Die Granularität der Abschnitte ist abhängig von der verwendeten Architektur. Unterschieden wird zwischen Satz-Ebene, Token-Ebene und Zeichen-Ebene.
Die Token-Ebene befindet sich zwischen der Satz- und Zeichen-Ebene und trennt Wörter in Teilwörter. Zum Beispiel wird *don't* in *do* und *n't* zerteilt. Auf diese Weise, kann für das Wort *doesn't*, zerteilt in *does n't*, das Wissen über *n't* wieder angewendet werden, ohne es neu zu erlernen. Der Vorteil die Texte auf Satz-Ebene zu trennen besteht darin, dass die Unterteilung einfach ist. Ein Vorteil der Token-Ebene gegenüber der Zeichen-Ebene ist, dass die Zeichen-Ebene oft zu feingranular ist und einzelne Zeichen wenig bis keine Information enthalten (bezogen auf lateinische Zeichen).

2.3 Machine Learning

Machine Learning ist ein Teilgebiet der künstlichen Intelligenz und bezieht sich auf die Fähigkeit von Maschinen, Muster in Daten zu erkennen und auf Grundlage dieser

Muster Vorhersagen und Entscheidungen zu treffen. Im Bereich des Machine Learnings lassen sich die Aufgaben in verschiedene Gruppen von Aufgaben unterteilen. Zu den häufigsten Gruppen gehören:

- **Classification** Gegeben ein Datenpunkt und einer Menge von Klassen. Die Aufgabe besteht darin, dem Datenpunkt eine passende Klasse zuzuweisen.
  Beispiel: *Es gibt ein Datensatz mit Tierbildern. Jedes dieser Bilder ist einer Klasse, einer Tierart zugeordnet. Ein neues Bild soll einen dieser Klassen zugeordnet werden.*

- **Regression** Gegeben eine Menge von Eingabedaten, den numerischen Zielwert für diese Daten bestimmen.
  Beispiel: *Es soll für ein Haus der Preis geschätzt werden. Zu diesem Zweck wird ein Datensatz mit verschiedenen Parametern verwendet, die relevant für die Bestimmung des Kaufpreises sind, wie beispielsweise die Fläche, das Baujahr oder die Lage des Objektes (Eingabedaten). Zusätzlich ist der tatsächliche Kaufpreis als Zielwert bekannt. Auf Basis dieser Daten soll dann für Eingabedaten eines Hauses der Kaufpreis geschätzt werden.*

- **Anomaly Detection** Das Ziel besteht darin zu erkennen, ob ein gegebener Datenpunkt ein normales Verhalten aufweist oder ungewöhnlich beziehungsweise abweichend ist. Anomaly Detection kann auch als ein binäres Klassifizierungsproblem interpretiert werden. Beispiel: *Für eine neue Kreditkartenabrechnung soll festgelegt werden, ob diese im Rahmen des üblichen Verhaltensmuster liegt oder auf einen möglichen Kreditkartendiebstahl hindeutet.*

Im Bereich des Machine Learning gibt es zudem eine weitere essentielle Unterscheidung in der Klasse von Algorithmen. Einmal in den Bereich des *supervised learning* und des *unsupervised learning*. Das *supervised learning*, auch bekannt als unterstütztes Lernen, setzt voraus, dass die korrekten Werte oder Labels für das Training der Algorithmen bekannt sind. Die Aufgabe der Algorithmen ist es, eine Funktion zu erlernen, die die Eingabedaten auf die korrekten Labels abbildet. Die Modelle können dann verwendet werden, um zukünftige Eingabedaten zu klassifizieren oder zu prognostizieren.
Im Kontext des *unsupervised learning* stehen keine Labels zur Verfügung und das Ziel besteht darin, Muster und Strukturen in den Eingabedaten automatisch zu identifizieren. Basierend auf den entdeckten Mustern können dann Aussagen für zukünftige Daten getroffen werden.

Zur Validierung von Machine-Learning-Modellen wird üblicherweise ein zweiteiliger Datensatz verwendet, der in einen Trainings- und einen Testdatensatz aufgeteilt wird. Dieser Schritt erfolgt vor dem eigentlichen Trainieren des Modells. Das Modell wird auf dem Trainingsdatensatz trainiert und anschließend auf den Daten des Testdatensatzes validiert. Diese Aufteilung stellt sicher, dass das Modell nicht zu stark auf spezifische Merkmale des Trainingsdatensatzes reagiert, sondern auch auf vorher unbekannten Daten generalisieren kann.
Um den Grad der Korrektheit zu messen, gibt es unterschiedliche Metriken, die für

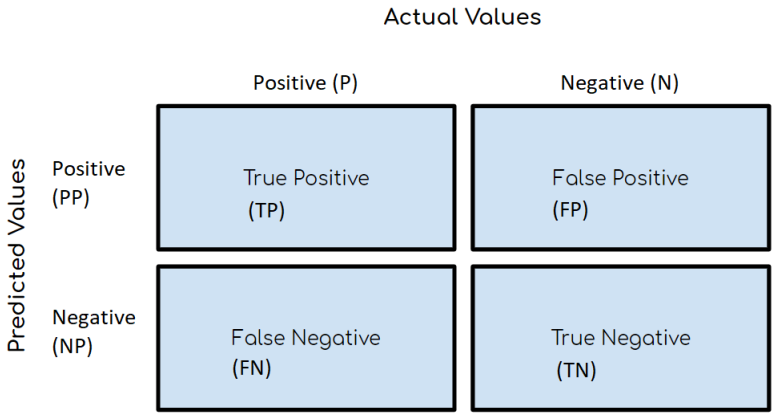

Abbildung 2.1: Confusion Matrix einer 2-Klassen Klassifizierung

die unterschiedlichen Klassen der Aufgaben geeignet sind.
Für eine 2-Klassen Klassifizierungsaufgabe lassen sich die Datenpunkte in vier Gruppen unterteilen, illustriert durch die *confusion matrix* in Abbildung 2.1. Für ein Klassifizierungsproblem ist die naheliegende Methode zur Validierung, die Anzahl korrekter Vorhersagen durch die Anzahl der gesamten Vorhersagen zu teilen

$$Accuracy = \frac{\#\, korrekter\, Vorhersagen}{\#\, gesamten\, Vorhersagen} \tag{2.1}$$

$$= \frac{TP + TN}{P + N} \tag{2.2}$$

Accuracy ist nur eine sinnvolle Metrik für eine Klassifizierung mit einer gleichmäßigen Verteilung an Klassen.
Um dieses Konzept zu verdeutlichen, kann ein Beispiel herangezogen werden. Ein Datensatz enthält 99% Datenpunkte der Klasse A und nur 1% der Klasse B. Ein Modell das unabhängig von der Eingabe Klasse A ausgibt erreicht eine Genauigkeit von 99%. Diese Metrik spiegelt jedoch nicht die tatsächliche Qualität des Modells wider, da es in dem Fall keine Unterscheidung der Klassen vornimmt.
Eine zusätzliche Metrik, welche im Rahmen der Evaluierung von Klassifikationsalgorithmen Verwendung findet, ist der F1-Score.

$$F_1 = 2\,\frac{precision \cdot recall}{precision + recall} \tag{2.3}$$

Der F1-Score stellt dabei ein Mittelwert zwischen den Metriken Precision und Recall dar.
Die Precision gibt den Anteil der korrekt positiv klassifizierten Fälle innerhalb der Gesamtzahl der als positiv klassifizierten Fälle an.

$$Precision = \frac{TP}{PP} \tag{2.4}$$

Der Recall hingegen gibt den Anteil der korrekt positiv klassifizierten Fälle innerhalb der Gesamtzahl der tatsächlich positiven Fälle an.

$$Recall = \frac{TP}{P} \tag{2.5}$$

Der F1-Score ist ein geeignetes Maß, wenn sowohl Precision als auch Recall gleichermaßen wichtig für die Anwendung sind. [30]
Für Regressionaufgaben wird zur Evaluierung die Abweichung von dem geschätzen Wert zu dem tatsächlichen Wert verwendet. Ein typisches Beispiel zum Messen der Abweichung ist der mean squared error (MSE)

$$MSE = \frac{1}{N}[\sum(\hat{Y} - Y)^2] \tag{2.6}$$

Anomaly Detection ist eine 2-Klassen Klassifizierungsaufgabe. Aus diesem Grund können die Metriken aus dem Abschnitt zu Klassifizierungsaufgaben auch für diese Aufgabe verwendet werden. Statt einer binären Ausgabe (Inlier, Outlier) geben Algorithmen einen Wert einer Metrik aus.
Ein Schwellwert kann verwendet werden, um die Datenpunkte mit einem kleineren Wert als Inlier und die Punkte mit einem höheren Wert als Outlier zu klassifizieren. Für diesen Schwellwert können die Metriken der Klassifizierung angewendet werden, um die Güte des Schwellwertes zu quantifizieren. Bei dem ROC-AUC-Score (*area under the curve*) [11] werden für eine Reihe (n>>10) von Schwellwerten die false-positive-rate (fpr) und die true-positive-rate (tpr) berechnet.

$$tpr = \frac{TP}{P},\ fpr = \frac{FP}{N} \tag{2.7}$$

Für jeden Schwellwert wird die fpr gegen die tpr abgebildet, die daraus resultierende Fläche unterhalb der Kurve wird als AUC-Score bezeichnet. Exemplarisch sind fünf Kurven in Abbildung 2.2 abgebildet.

In der folgenden Arbeit bezieht sich der AUC-Score auf den ROC-AUC-Score.
Der AUC-Score eines Klassifiziers lässt sich als die Wahrscheinlichkeit interpretieren, dass der Klassifizierer ein zufälligen positven Datenpunkt höher stuft, als ein zufälligen negativen Datenpunkt.

AUC Score. *Gegeben ein Klassifizierer k, und ein Datensatz D mit positiven und negativen Elementen. Sei $x^+ \in D$ ein zufälliges Element der positiven Klasse und $x^- \in D$ ein zufälliges Element der negativen Klasse.*

$$AUC = P(score(x^+) > score(x^-)) \tag{2.8}$$

Der AUC-Wert liegt im Bereich von 0 bis 1. Ein Wert von 0,5 bedeutet, dass das Modell nicht besser als eine zufällige Vorhersage ist.
Ein AUC-Wert von 0 deutet darauf hin, dass das Modell eine vollständig falsche Klassifikation durchführt. Es trennt die Klassen entgegengesetzt zu den tatsächlichen Werten. Inlier werden als Outlier klassifiziert und Outlier als Inlier.

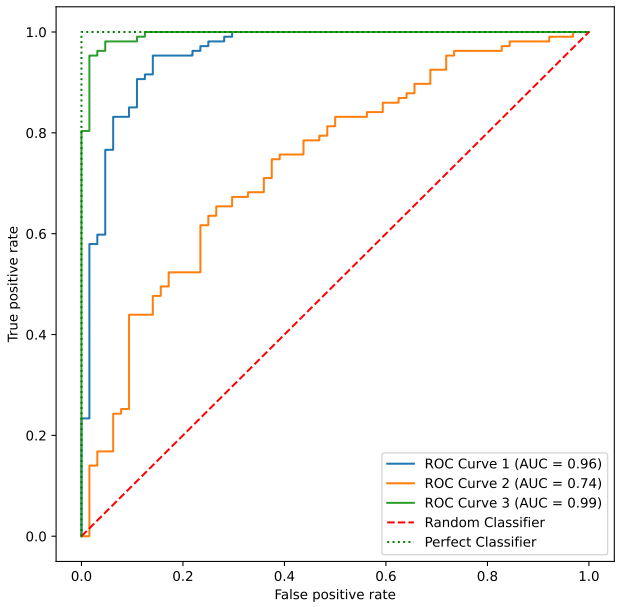

Abbildung 2.2: Verschiedene ROC-Curves abgebildet. Grün: AUC=0.99, Blau: AUC=0.96, Orange: AUC=0.74, Rot: AUC=0.5 (Zufälliges auswählen einer Klasse), grau: AUC=1 (perfektes Ergebnis). Quelle: [2]

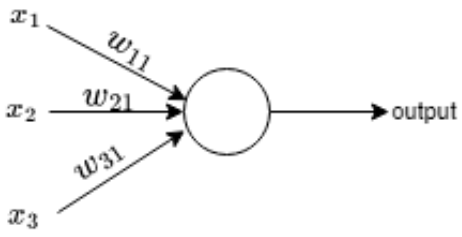

Abbildung 2.3: Einlagiges Perzeptron

Ein AUC-Wert von 1 zeigt an, dass das Modell eine perfekte Klassifikation durchführt und die Klassen vollständig trennt.

Eine sehr relevante Machine Learning Architektur bilden die Neuronalen Netzwerke. Ein Neuronales Netzwerk ist auf abstrakter Ebene eine Funktion f, die eine Eingabe x in eine Ausgabe y projiziert $y = f(x)$. Die einfachste Form eines Neuronalen Netzwerkes ist das einlagige Perzeptron. Eine Darstellung befindet sich in Abbildung 2.3. Das Ergebnis eines Neurons ist die gewichtete Summe der vorherigen Neuronen und einem Bias b_i.

$$y = f(x_1, x_2, x_3) = x_1 * w_{11} + x_2 * w_{2_1} + x_3 * w_{3_1} \tag{2.9}$$

Die Funktion ist linear. Um komplexere Funktionen zu erlernen, ist es notwendig eine nichtlineare Aktivierungsfunktion φ zu verwenden, damit das Netzwerk nichtlineare Zusammenhänge zwischen der Eingabe und der Ausgabe erlernen kann.

$$y = f(x_1, x_2, x_3) = \varphi(x_1 * w_{11} + x_2 * w_{2_1} + x_3 * w_{3_1}) \tag{2.10}$$

Zwei typische Beispiele für eine Aktivierungsfunktion sind die ReLU- und die Sigmoid-Aktivierungsfunktion.

$$Sigmoid: \ \varphi(x) = \frac{1}{1 + e^{-x}} \tag{2.11}$$

$$ReLU: \ \varphi(x) = max(0, x) \tag{2.12}$$

Die Aufgabe des Bias in Neuronen ist es, eine Verschiebung der Aktivierungsfunktion zu erzeugen und somit die Aktivierung des Neurons zu steuern.

Neuronale Netzwerke bestehen aus vielen miteinander verbundenen Neuronen, die in Schichten (*Layer*) organisiert sind. Diese Schichten werden, wie in Abbildung 2.4 dargestellt, in der Regel in drei Haupttypen unterteilt, den Input-Layer, den Hidden-Layer und den Output-Layer. Der Input-Layer ist verantwortlich für das Einlesen von Datenpunkten, die als Eingabe in das neuronale Netzwerk dienen. Im Falle von multivariaten Datenpunkten erfordert das eine Anzahl von n Neuronen, wobei n der Anzahl an Dimensionen des Datenpunktes entspricht. Die Werte werden an den Hidden-Layer weitergegeben und in jedem Neuron die Ausgabe berechnet. Anschließend wird die Ausgabe weiter zum Output-Layer geleitet. Die Anzahl der

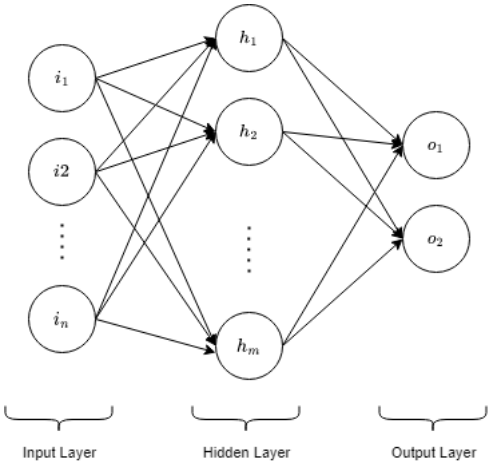

Abbildung 2.4: Beispiel Neuronales Netzwerk mit n Inputs, einem hidden layer mit m Neuronen und 2 Neuronen im Output Layer

Neuronen im Output-Layer ist abhängig von der Art der Aufgabe. Bei einer Anzahl von k Klassen oder bei einer k-dimensionalen Regressionsaufgabe werden k Neuronen benötigt.
Auf Basis der Neuronalen Netzwerke gibt es viele verschiedene Architekturen, die für verschiedene Anwendungen benutzt werden. Bei der dargestellten Architektur in der Abbildung 2.4 handelt es sich beispielsweise um ein feed-forward network, mit einer Reihe von komplett verbundenen Schichten.
In einem Training schritt des Neuronalen Netzwerks wird zuerst ein Datenpunkt durch das Netzwerk iteriert. Anschließend wird der Fehler berechnet, der Auskunft über die Genauigkeit des Modells gibt. Hierfür wird eine Verlustfunktion verwendet, die die Abweichung zwischen der vom Netzwerk geschätzten Ausgabe und der tatsächlichen Ausgabe berechnet. Das Neuronale Netzwerk lernt, in dem die Gewichte angepasst werden, so dass die Verlustfunktion minimiert wird und die geschätzten Ausgaben sich den tatsächlichen Werte nähern.

2.4 Deep Learning

Deep Learning ist ein Teilgebiet des Machine Learnings. Machine-Learning-Algorithmen basieren auf statistischen Modelle, die es ermöglichen Vorhersagen zu treffen. Im Bereich des Deep Learnings werden tiefe Neuronale Netzwerke (Deep NN) verwendet, mit mehren hidden Layer verwendet, die in der Lage sind komplexe Funktionen abzubilden und zu erlernen.

Deep Neural Network. *Sei h_i die Ausgabe des i-ten hidden layer, i die Ausgabe des input layer und o die Ausgabe des Output Layer. Dann ergibt sich die Ausgabe eines Deep NN f mit 3 hidden layer bei der Eingabe e durch*

$$f(e) = o(h_3(h_2(h_1(i(e))))) \tag{2.13}$$

Das Minimum der Verlustfunktion analytisch zu bestimmen ist bei einer hohen Anzahl von Variablen nicht möglich, da die Verlustfunktionen von tiefen Neuronalen

Netzwerken von Millionen von Gewichten abhängen.
Die Gewichte des Neuronalen Netzwerkes werden stattdessen durch den Gradientenabstieg angepasst. Der Gradientenabstieg ist ein Optimierungsverfahren mit dem Ziel eine Funktion zu minimieren, indem der Wert der Funktion schrittweise reduziert wird. Der Gradient einer Funktion C ist gegeben durch

$$\nabla C \equiv (\frac{\partial C}{\partial v_1}, \cdots, \frac{\partial C}{\partial v_m}) \tag{2.14}$$

Der Gradient beschreibt den Vektor, der aus den partiellen Ableitungen der Funk-

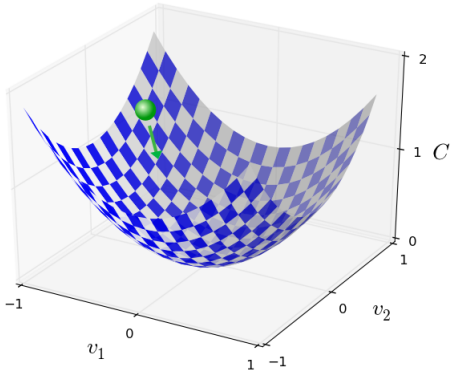

Abbildung 2.5: Gradientenabstieg symbolisiert durch eine Kugel an der Neigung eines Tales. [21]

tion C nach jeder Variablen entsteht. Die partiellen Ableitungen geben an, wie sich der Wert der Funktion in Richtung jeder Variablen ändert. Der Gradient zeigt in die Richtung der maximalen Steigung in einem Punkt.
Die Veränderung von C ist etwa proportional zum Gradienten von C in Richtung der Änderung in den Variablen $\triangle v$

$$\triangle C \approx \nabla C \cdot \triangle v \tag{2.15}$$

Um die Funktion C zu minimieren, muss die Veränderung der Variablen in die Gegenrichtung des Gradienten von C zeigen. Das Verfahren ist in Abbildung 2.5 symbolisch dargestellt.

$$\triangle v = -\alpha \nabla C \tag{2.16}$$

Der Parameter $\alpha << 1$, auch Lernrate, bestimmt bei dem Training von Neuronalen Netzwerken die Schrittweite bei der Anpassung des Parameters v. Gemäß Gleichung 2.15 ergibt sich, dass $\Delta C \approx -\alpha \nabla C \cdot \nabla C = -\alpha |\nabla C|^2$ immer abnimmt.

Im Kontext von neuronalen Netzwerken werden die Gewichte so angepasst, dass sie in entgegengesetzter Richtung des Gradienten der Verlustfunktion verändert werden. Auf diese Weise wird versucht, die Verlustfunktion zu minimieren.

$$w_{ij} \rightarrow w'_{ij} = w_{ij} - \alpha \frac{\partial C}{\partial w_{ij}} \tag{2.17}$$

Die Gewichte hier werden *online* angepasst. *Online-Learning* bezieht sich darauf, dass die Gewichte für jeden einzelnen Datenpunkt angepasst werden. Bei dem stochastischen Gradientenabstieg werden die Gewichte im *mini-batch* Verfahren angepasst. Dabei werden die Gewichte nicht direkt angepasst, sondern der Datensatz in mehrere Teilmengen (*mini-batches*) unterteilt und für jede Teilmenge die Gewichte in die gemittelte Richtung angepasst.
Die Neuronalen Netzwerken haben eine wichtige Rolle bei der Identifizierung von komplexen Funktionen in hochdimensionalen Daten. Die Neuronalen Netzwerken wurden größer und verschiedene Architekturen von Neuronalen Netzwerken etablierten sich.
Der Autoencoder [27] ist ein Beispiel einer solchen Architektur eines Neuronales Netzwerk bestehend aus einer Encoder-Decoder Architektur. Ein Encoder e transformiert die Eingabe in eine interne niedrigdimensionale Vektorrepräsentation. Der Decoder d transformiert diese Vektorrepräsentation zurück auf die Eingabe.

$$x = d(e(x)) \tag{2.18}$$

Das Ziel ist es die Eingaben abzubilden. Autoencoder werden in der Dimensionsreduktion und im Bereich der Anomalieerkennung verwendet. Für eine Dimensionsreduktion wird der Autoencoder auf den zu reduzierenden Daten trainiert, anschließend wird der trainierte Encoder verwendet, um die Eingaben in die niedrigdimensionale Repräsentation zu transformieren. Die Funktionsweise von Autoencoer im Bereich der Anomalieerkennung wird im nachfolgenden Kapitel beschrieben.

2.4.1 Anomaly Detection Methoden

Nachfolgend werden verschiedene Methoden zur Anomalieerkennung vorgestellt. Dabei beschränkt sich die Auswahl auf die in der vorliegenden Arbeit verwendeten Methoden.

Zur Anwendung der Algorithmen werden verschiedene Annahmen über die Anomalien getroffen. Anomale Datenpunkte sind verschieden von den normalen Datenpunkten und treten selten auf. Des weiteren wird die Annahme getroffen, dass normale Datenpunkte eine höhere lokale Dichte aufweisen, als Anomalien.

Der Isolation Forest [16] isoliert Anomalien, anstatt die Charakteristik normaler Punkte abzubilden. Für die Isolierung werden binäre Entscheidungsbäume verwendet. Die Entscheidungsbäume teilen die Datenmenge an jedem Knoten in einer zufälligen Dimension mit einem zufälligen Schwellwert auf. Der Isolation Forest verwendet ein Ensemble aus binären Entscheidungsbäumen, um Anomalien zu detektieren. Anomalien sind Datenpunkte mit einer kurzen durchschnittlichen Pfadlänge. Die Pfadlänge eines Datenpunktes in einem Baum berechnet sich aus der Anzahl an Kanten von der Wurzel zu dem Datenpunkt.
Die durchschnittliche Pfadlänge ergibt sich aus dem normierten Durchschnitt der Pfadlängen der Entscheidungsbäume.
Der Anomaliescore s eines Datenpunktes x in einem Datensatz mit n Instanzen ist

gegeben durch

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$$

mit der durchschnittlichen erwarteten Höhe $E(h(x))$, approximiert durch die durchschnittliche Höhe der Entscheidungsbäume. $c(n)$ ist der Durchschnitt von $h(x)$ über die n Instanzen und wird als Normierung verwendet. Anomalien sind "few and different" [16]. Damit wird angenommen, dass die Anzahl Anomalien gering ist und diese sich in den Attributen unterscheiden zu den normalen Datenpunkten. Diese beiden Eigenschaften resultiert in einer einfacheren Separierung der Anomalien und führen zu kürzeren durchschnittlichen Pfadlänge.

Ein weiterer Algorithmus ist der KNN (*k-Nearest-Neighbor*) [9]. Das Verfahren basiert auf dem Konzept der lokalen Dichte. Das bedeutet es klassifiziert Datenpunkte als Anomalie, basierend auf der Dichte des lokalen Bereichs um den Datenpunkt. Dabei werden Punkte mit einer höheren Dichte als Inlier klassifiziert. Zur Bestimmung der lokalen Dichte wird die Distanz zu den k nächsten Nachbarn des Datenpunktes gemittelt. Auf Basis dieser durchschnittlichen Entfernung und einem Schwellwert wird der Datenpunkt als Anomalie klassifiziert.

Wie im Kapitel 2.4 beschrieben, kann die Autoencoder Architektur auch für die Anomalieerkennung genutzt werden [28]. Der Autoencoder wird zuerst mit normalen Daten trainiert, um eine kompakte Darstellung der normalen Daten zu erlernen. Im Anschluss erfolgt die Anwendung des Autoencoders auf neue Daten. Der Rekonstruktionsfehler, definiert als die Distanz zwischen dem Originaldatenpunkt und dem rekonstruierten Punkt, dient als Maß zur Bestimmung von Anomalien. Der Autoencoder ist speziell darauf trainiert, normale Daten optimal abzubilden, wodurch eine bessere Abbildung normaler Daten erreicht wird. Hingegen weisen Anomalien einen vergleichsweise hohen Fehler auf. Daraus ergibt sich die Möglichkeit, den Rekonstruktionsfehler mithilfe eines Schwellwertes zur Klassifizierung zu verwenden.

2.4.2 Large Language Models

Dieses Kapitel gibt einen Überblick über large language models (LLMs). LLMs sind große Neuronale Netzwerke, die verwendet werden, um natürliche Sprache zu analysieren, zu verstehen und zu erzeugen. Typischerweise basieren LLMs auf der Transformer-Architektur auf die nachfolgend eingegangen wird.
Die Transformer-Architektur wurde erstmals im Paper „Attention is all you need" vorgestellt und hat seitdem eine Reihe von Weiterentwicklungen erfahren. Abbildung 2.6 zeigt die Architektur bestehend aus zwei Hauptkomponenten: Dem Encoder und dem Decoder. In der Regel sind die Eingabe- und Ausgabedaten eines Transformer-Modells natürliche Texte. Der Encoder-Block wandelt die Eingabe in eine interne Vektorrepräsentation um, während der Decoder-Block diese interne Repräsentation in die gewünschte Ausgabe transformiert. In einem Szenario, wie einer Englisch-Französisch-Übersetzung, wäre die Eingabe der englische Satz, die interne Vektorrepräsentation würde eine numerische Darstellung enthalten, die die Information des Satzes zusammenfasst und die Ausgabe wäre der entsprechende französische

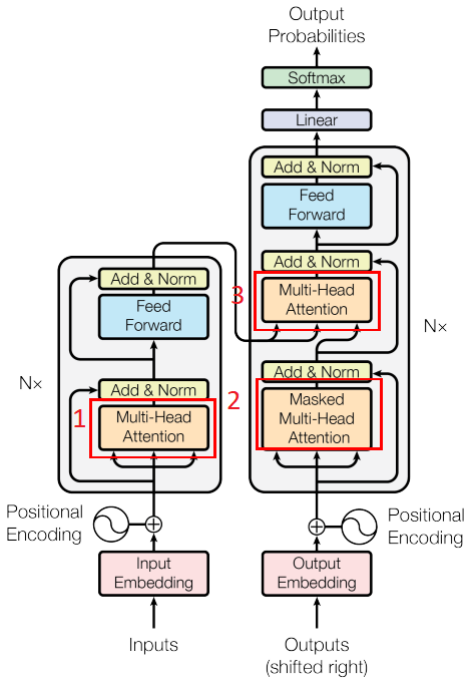

Abbildung 2.6: Transformer Architektur aus dem Paper "Attention is all you need". Rot hervorgehoben sind die 3 Attention Blöcke

Satz.
Die Transformer Architektur ist ein *autoregressive* Modell. Die zukünftigen Ausgaben basieren auf vorherigen Werten. Die Sätze haben eine zeitlich korrelierte Komponente, da jedes Wort s zum Zeitpunkt t (Position t im Satz) von den vorherigen Zeitpunkten abhängt.

$$p(x) = \prod_{i}^{t} p(s_t|s_1, \ldots, s_{t-1}) \tag{2.19}$$

Die Ausgabe des Decoders ist gegeben durch die Eingabe und die bis zu dem Zeitpunkt generierte Ausgabe. Das Modell generiert ein Wort in einer Iteration. Die Vorgehensweise wird durch ein Beispiel deutlicher. Um das oben beschriebene Szenario aufzugreifen: gegeben den zu übersetzenden englischen Satz 'How are you' als Eingabe, ergibt sich die resultierende französische Übersetzung:

- *Iteration 1*: Gegeben "", das Modell schätzt als nächstes Wort 'Comment'
- *Iteration 2*: Gegeben 'Comment', das Modell schätzt als nächstes Wort 'ça'
- *Iteration 3*: Gegeben 'Comment ça', das Modell schätzt als nächstes Wort 'va'
- *Iteration 4*: Gegeben 'Comment ça va', das Modell schätzt als nächstes Wort '?'
- *Iteration 5*: Gegeben 'Comment ça va ?', das Modell schätzt als nächstes Wort '$< END >$' und signalisiert das Ende der Ausgabe

Das Ziel der Transformer-Architektur ist es, sowohl den Encoder als auch den Decoder zu optimieren, um eine bessere Repräsentation der Eingaben zu erzielen und bessere Ausgaben auf Basis der Repräsentationen zu generieren.

Vor der Einführung der Transformer-Architektur galten Recurrent Neural Networks (RNNs) als State-of-the-art-Modell für zeitlich korrelierte Sequenzen. Im Gegensatz zu herkömmlichen Neuronalen Netzwerken haben RNNs die Fähigkeit, diese zeitliche Abhängigkeit von Eingabedaten zu erfassen, indem sie Feedback-Schleifen einbeziehen.
Obwohl RNNs in der Lage sind, die zeitliche Abhängigkeit von Eingabedaten zu erfassen, haben sie einige, die Leistung betreffende Einschränkungen. Ein Hauptproblem besteht darin, dass lange Abhängigkeiten in Texten in der Regel nicht aufrecht erhalten werden können. Der Grund dafür ist die sequentielle Natur der Feedback-Schleifen. Diese verursacht, dass die Gradienten während des Trainings entweder explodieren, oder verschwinden können. Ein weiteres Problem der sequentiellen Charakteristik ist, dass Eingaben nicht als Ganzes, sondern sequentiell eingegeben werden müssen. Das verlangsamt sowohl den Trainingsprozess, als auch den späteren Anwendungsfall.

RNNs wurden weiterentwickelt, um diese Hauptprobleme zu adressieren. Eine Unterform des RNN sind die LSTMs (Long Short-Term Memory), die durch ihre Architektur das Problem der Gradienten teilweise lösen konnten.
Anders als die RNNs benutzen Transformer nicht rekursive Verbindungen, um Worten einen Kontext zu geben, sondern verwenden den sogenannten Attention Mechanismus. Der Attention-Mechanismus ermöglicht die effektive Extraktion relevanter Informationen aus der Eingabe, um eine adäquate Generierung von Ausgaben zu ermöglichen. Die Eingabe erfolgt dabei nicht mehr sequentiell, sondern parallel. Um einem Wort eine relative Position zu geben, wird das positional encoding verwendet. Die Position ist wichtig, denn die Bedeutung eines Satzes hängt von der Reihenfolge der Wörter im Satz ab.
Der Attention-Mechanismus dient dazu, die Beziehung zwischen Wörtern zu gewichten und so die Semantik des Satzes besser zu verstehen. Der Transformer enthält drei verschiedene Attention-Blöcke (Abbildung2.6). Der Attention-Block besteht aus drei Eingaben: Query (Q), Keys (K) und Values (V). Q,K,V sowie die Ausgabe sind Vektoren.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.20}$$

Die Attention Blöcke 1 und 2 sind self-attention Blöcke, die Teile der Eingabe, basierend auf den Eingaben selbst, gewichten. Der Attention-Block 3 hingegen agiert als Filter, welcher die Eingabe auf Basis der bisher generierten Ausgabe gewichtet.
Zur Erläuterung des Attention-Mechanismus wird nachfolgend die semantische Bedeutung der einzelnen Komponenten und Operationen beschrieben:
Die Query stammt aus der bisherigen Ausgabe und der Key und Value aus der Eingabe. Um die Relevanz der Werte in Bezug auf die Abfragen zu bestimmen, wird das Skalarprodukt zwischen den Vektoren Q und K^T berechnet, wobei der Wert größer

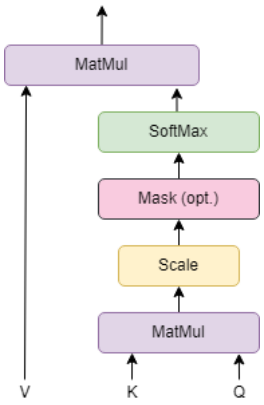

Abbildung 2.7: Die Attention Berechnung in Gleichung 2.20 anschaulich visualisiert. Abwandlung aus der Arbeit [33]

wird, je ähnlicher die beiden Vektoren sind. Die SoftMax Funktion transformiert die Skalarprodukte in eine Verteilung, die zwischen 0 und 1 liegt und deren Summe 1 ergibt. Die Exponentialfunktion in der SoftMax-Funktion sorgt dafür, dass größere Werte in der Verteilung einen höheren Anteil erhalten. Die Ausgabe der SoftMax-Funktion kann als Index interpretiert werden, der einen Vektor mit einem Eintrag von 1 an der Position i und allen anderen Einträgen nahe 0 enthält. Dies führt dazu, dass bei der Matrixmultiplikation der Wert an der Position V_i ausgewählt wird.

Das LLM GPT2 [26] und GPT3 [5] sind populäre Beispiele, die auf der Basis der Transformer-Architektur basieren.
Das GPT2- und GPT3-Modell sind *decoder only* Architekturen. Das Modell verwendet nur den Decoder Block des Transformers. Der Attention Mechanismus kann nur auf vorherige Werte zugreifen (*masked attention*). Aus diesem Grund wird die Architektur oft auch als *auto-regressive models* bezeichnet.
Die Eingabe wird durch ein Spezialtoken von der bisher generierten Ausgabe getrennt.

Die Architekturen unterscheiden sich nur in kleinen Anpassung von der originalen Transformer Architektur. Stattdessen, wird die erhöhte Leistungsfähigkeit erreicht, indem große Modelle (175 Milliarden Parameter im Falle von GPT3) auf riesigen Datenmengen (300 Milliarden Token) trainiert werden.

Die *self-supervised* Charakteristik ermöglicht es diese riesigen Trainingsdatensätze zu erstellen. Das Ziel der Architektur ist das nächste Wort zu schätzen. Für gegebene Textdaten ist das nächste Wort bekannt:
Der gegebene Satz

A robot may not injure a human being or, through inaction, allow a human being to come to harm. (Erstes Gesetz der Robotik)

kann verwendet werden als Trainingsdatenpunkt durch

$X = A\ robot\ may\ not\ injure\ a\ human\ being\ or,\ through\ inaction,\ allow\ a\ human\ being\ to\ come\ t$
$y = harm$

Die Konstruktion der großen Datensätze ohne Erstellung von Labels ermöglicht das Trainieren von großen Modellen, die aus mehreren verbundenen Decoder-Blöcken bestehen, die in Schichten organisiert sind (96 für GPT3).

Die Architektur ist nicht für eine spezifische Aufgabe trainiert, sondern eine *multi-task* Architektur, die spezifische Anwendung ergibt sich aus der Eingabe. Als Beispiel wird in der Arbeit [26] erwähnt, dass die Architektur für das Zusammenfassen eines Textes verwendet werden kann, ohne das Trainieren der Aufgabe, durch das Hinzufügen des Textes "TL;DR" an das Ende des zusammenzufassenden Textes.

3 Related Work

In diesem Abschnitt werden verwandte Arbeiten im Bereich der Anomalie Erkennung vorgestellt. Während Anomalieerkennung ein gut erforschtes Thema in der wissenschaftlichen Literatur ist, sind Arbeiten im Thema Anomalieerkennung auf Text Daten wenig verbreitet. Die Arbeiten zeigen, wie in verschiedenen Ansätzen die Anomalieerkennung auf Textdaten erzielt wird.

Die Arbeit "Anomaly Detection in Text Data Sets using Character-Level Representation" [18] transformiert in einem ersten Schritt die Texteingaben in eine Vektorrepräsentation fixer Länge. Dazu wird der Text zuerst in eine Sequenz von Zeichen unterteilt. Die einzelnen Zeichen sind durch ein vordefiniertes Alphabet gegeben und jedes Zeichen wird durch den Index in dem zufällig angeordneten Alphabet repräsentiert. Zeichen die nicht im Alphabet sind, werden als 0 codiert. Das Modell transformiert die Sequenz von Zeichen in einen Vektor mit einer fixer Länge, die der durchschnittlichen maximalen Länge der Textmerkmale entspricht. Jedes Zeichen, das über diese Länge hinausgeht, wird ignoriert.
In einem nächsten Schritt werden die Vektoren mit einem Ensemble aus verschiedenen Anomalie Klassifikatoren klassifiziert. Die Autoren verwenden PCA, COPOD, HBOS, LODA , CBLOF und IForest [29] [15] [7] [24] [35] [16]. Die Anomalyscores der einzelnen Klassifikatoren werden mit dem AOM- (Average over Maximum) bzw. MOA-Score (Maximum over Average) [1] zu einem Gesamtergebnis kombiniert.
Das Paper wendet das Verfahren auf einem privaten Datensatz und dem öffentlichen SMS-Spam Datensatz [32] an. Auf dem SMS-Spam Datensatz erreicht das Paper einen AUC-Score von 87%.

Die Arbeit von Jafari et al. [12] beschreibt eine alternative Methode zur Erstellung von Repräsenationen von Textdaten unter Verwendung von SBERT [22]. Das Sprachmodell arbeitet auf einem Satzlevel und erfordert daher in einem ersten Vorverarbeitungsschritt, die Unterteilung der Texte in einzelne Sätze. Die supervised Architektur der Arbeit besteht aus einem Convolutional Autoencoder, der auf den unterteilten Texten (in Sätzen) trainiert wird. Jeder Text wird dabei als 3-dimensionales Array in den Autoencoder eingegeben (1, `max_sent`, `feature_dim`). Die Anzahl an Feature Dimensionen (`feature_dim`) ist gegeben durch das SBERT Modell mit 768 und die maximale Anzahl an Sätzen pro Text (`max_sent`) setzen die Autoren auf einen fixen Wert. Das Ziel des Autoencoders ist die Dimensionsreduktion der Sequenz von Sätzen. Sowohl der reduzierte Vektor als auch der Rekonstruktionsversuch des Autoencoders werden in eine logistische Regression eingespeist, um zwischen anomalen und normalen Datenpunkten zu klassifizieren.
Die Autoren verwenden den XNLI Korpus als Kerndatensatz. Zusätzlich nutzen sie

als Hilfsdatensatz den Standford Sentinment Treebank (sst) Korpus. Aus dem sst Datensatz werden 1000 Samples ausgewählt und in den XNLI Datensatz injiziert. Auf diese Weise entsteht ein Datensatz, in dem die XNLI Samples normal sind und die sst Samples Anomalien darstellen. Das Modell erreicht einen F1-Score von 0.86.

Das Paper von Mahapatra et al. [17] stellt eine Methode zur Anomalieerkennung in Textdaten vor, die auf einer Kombination von Content Analysis und Context Analysis basiert. In der Content Analysis werden Anomalien basierend auf den vorhandenen Daten bestimmt. In der Context Analysis werden diese Anomalien in einer zweiten Instanz gegen einen externen Textkorpora validiert, um zu erkennen, ob die Themen auch in einem generellen Kontext, also außerhalb der vorliegenden Daten, anomal sind.
In der Content Analysis wird in einem ersten Schritt die Latent Dirichlet Allocation (LDA) [4] verwendet, um Themen aus den Dokumenten zu extrahieren. LDA verwendet collapsed Gibbs sampling [13] um die Themen iterativ zu approximieren. Jedes Dokument ist eine Verteilung über die Themen und jedes Thema eine Verteilung über dem Wortkorpus.
Das Ziel besteht darin, die Themen-Wort Zuordnung zu berechnen, die jedem Wort in jedem Dokument einem Thema zuordnet. Aus der Verteilung der Dokumente über den Wörtern und damit den Themen lassen sich die zugehörigen Themen der Dokumente extrahieren. Die Themen-Wort Verteilung lässt sich nicht analytisch berechnen, weshalb das collapsed Gibbs Sampling eingesetzt wird, um die unterliegende Verteilung approximativ zu berechnen.
In der LDA wird initial jedem Wort in jedem Dokument ein zufälliges Thema zugeordnet. Anschließend wird über jedes Wort in jedem Dokument iteriert und folgendes berechnet

- P(Thema t | Dokument d): Die Wahrscheinlichkeit, dass Thema t in Dokument d vorkommt. Gegeben durch das Verhältnis von Wörter in Dokument d, die aktuell Thema t zugeordnet sind.

- P(Wort w | Thema t): Die Wahrscheinlichkeit, dass Wort w unter dem Thema t generiert wurde. Gegeben durch die Anzahl Zuordnungen von Wort w zum Thema t, im Verhältnis zu der gesamten Anzahl Zuordnungen zum Thema t.

- P(Thema t, Wort w | Dokument d): Die Wahrscheinlichkeit, dass Wort w dem Thema t im Dokument d zugeordnet wird. Die Wahrscheinlichkeit ergibt sich (mit der Annahme der Unabhängigkeit der beiden zuvor gerechneten Wahrscheinlichkeiten) aus P(Thema t | Dokument d) $*$ P(Wort w | Thema t). Mit der Wahrscheinlichkeit wird w dem (neuen) Thema t zugeordnet

Der Vorgang wird für eine bestimmte Anzahl an Iterationen wiederholt. Die extrahierten Themen werden anschließend absteigend nach ihrer gemeinsamen Verwendungen in den Dokumenten geordnet. Der niedrigste Teil, der im Paper mit 1/3 festgelegt wird, wird als anomale Themen deklariert. Die anomalen Themen werden anschließend in der Context Analysis mit den normalen Themen verglichen, um

einen *context score* zu berechnen. Dieser score ist die Summe der semantischen Distanz von den Wörtern des anomalen Themas zu allen anderen normalen Themen. Zur Berechnung der semantischen Distanz benutzt das Paper WordNet und Google. Kleinere Werte deuten auf eine höhere semantische Beziehung zwischen zwei Wörtern. Die anomalen Themen werden aufsteigend sortiert und der untere Teil als anomal deklariert. Themen die im ersten und zweiten Schritt als anomal deklariert wurden, werden insgesamt als abnormal markiert.
Die Arbeit zeigt, das durch die Kontext Analyse der F1 Score gestiegen ist, durch sinkende false positive rate. Des weiteren verwenden die Autoren das Konzept der Kontext Analyse auf Tags von YouTube Videos und zeigen, dass in einem weiteren Schritt auch anomale Wörter in Themen gefunden werden können.

Das Paper "Deep Anomaly Detection with Outlier Exposure"[6] von Dan Hendrycks et al. präsentiert eine neue Methode der Anomalieerkennung, die als Outlier Exposure (OE) bezeichnet wird. Im Gegensatz zu den meisten bestehenden Arbeiten, die Anomaly Detection im unsupervised Rahmen anwenden, nutzt OE eine nahezu unbegrenzte Menge an zufälligen natürlichen Bildern oder Texten, um abweichende Datenpunkte zu klassifizieren.

OE nutzt die Tatsache, dass das Internet für Anomalieerkennungsprobleme auf Bildern oder Texten eine fast unbegrenzte Menge an natürlichen Bildern oder Texten enthält, von denen anzunehmen ist, dass sie von den normalen Datenpunkten abweichen. Die Autoren schlagen daher vor, diese Daten als OE zu nutzen, um das Modell auf eine breitere Palette von Anomalien vorzubereiten und damit die Erkennungsleistung zu verbessern.

Die OE-Methode wurde auf einem normalen Datensatz trainiert, der um OE-Daten erweitert wurde und Anomalien erkennen soll. Der konservative Algorithmus wurde dann eingesetzt, um aus den vorhandenen Anomalien auf bis dahin unbekannte Anomalien zu schließen und diese zu erkennen. Die Autoren führten umfangreiche NLP Experimente, sowie kleinen und großen Vision-Aufgaben durch. Dabei stellten sie fest, dass OE die Erkennungsleistung von den Basismodellen signifikant verbesserte und besonders die false positive rate durch das Hinzufügen von Outliern reduziert wurde.

Die Verwendung von OE hat das Potenzial, die Anomalieerkennung in verschiedenen Bereichen zu verbessern, indem sie das Modell auf eine breitere Palette von Anomalien vorbereitet. Die Methode hat jedoch auch einige Herausforderungen, wie zum Beispiel die Auswahl geeigneter OE-Daten und die Vermeidung von Überanpassung an OE-Daten.

Insgesamt stellt das Paper *Deep Anomaly Detection with Outlier Exposure* einen wichtigen Beitrag zur Anomalieerkennung dar. Es zeigt, dass die Verwendung von OE eine vielversprechende Methode ist, um die Erkennungsleistung zu verbessern, insbesondere in Bereichen, in denen es eine breite Palette von Anomalien gibt. Es ist jedoch wichtig, dass weitere Forschung durchgeführt wird, um die Herausforderungen bei der Verwendung von OE zu adressieren und die Methode weiter zu verbessern.

Die Arbeit von Liznerski et al. [25] adaptiert die Idee des OE. Die Autoren stellen einen unsupervised OE Algorithmus vor, der eine Modifikation des *Deep Semi-supervised Anomaly Detection* (Deep SAD) Algorithmus ist, die sie *Hypersphere Classification* (HSC) nennen.
Deep SAD erfordert, dass sowohl normale als auch einige bekannte anomale Datenpunkte vorhanden sind. Der Algorithmus trainiert ein Netzwerk ϕ_θ, das normale Datenpunkte nahe einem Zentrum c konzentriert, während anomale Datenpunkte weg vom Zentrum projiziert werden.
In Deep SAD wird davon ausgegangen, dass diese bekannten Anomalien durch domain-spezifisches Wissen von Experten oder Ähnlichem gegeben sind. Im Gegensatz dazu verwenden die Autoren für HSC den OE-Hilfsdatensatz als Anomalien.
Die Autoren verwenden in HSC einen Cross-Entropy-Loss und geben an, dass diese Anpassung die Performance des Algorithmus im Vergleich zu Deep SAD erhöht. Zusätzlich verwenden die Autoren eine Radial-Basis-Funktion $l(z) = exp(- \parallel z \parallel^2)$, um die Darstellung von normalen Punkten kompakter zu gestalten. Die resultierende Zielsetzung ergibt sich durch:

$$\frac{1}{n}\sum_{i=1}^{n} y_i \parallel \phi_\theta(x_i) \parallel^2 -(1-y_i)log(1-ep(- \parallel \psi_\theta(x_i) \parallel^2))$$

Für einen gegebenen Datensatz $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, mit $x_i \in \mathbb{R}^d$ und $y \in 0, 1$. y ist das Label des Datenpunktes und gegeben durch $y = 1$ bei einem normalen und $y = 0$ einem anomalen Datenpunkt.

In den vorliegenden Arbeiten werden unterschiedliche Ansätze zur Anomalieerkennung auf Textdaten untersucht und vorgestellt. Die Methoden und Grundlagen der untersuchten Arbeiten sind dabei der vorliegenden Arbeit klar abgegrenzt und bieten somit einen wichtigen Rahmen für die weiterführende Forschung. Die verwandten Arbeiten zeigen auch, dass es nur eine andere Arbeit mit Large Language Models gibt, die jedoch supervised ist. Large Language Models sind ein wichtiges Forschungsthema und zeigten in vielen anderen Applikationen bereits state-of-the-art performance. Aus diesem Grund liefert die vorliegende Arbeit wichtige Beiträge zu dem interdisziplinären Forschungsgebiet der Anomalieerkennung und NLP.

4 Methodik und Implementierung

In diesem Kapitel wird die Methodik der Arbeit beschrieben.

Das untersuchte Modell besteht aus zwei Komponenten (Abbildung 4.1). Die erste Komponente umfasst die Extrahierung von Features aus den Texten und die Erzeugung eines Featurevektors. Die zweite Komponente beinhaltet die Anomalieerkennung auf den extrahierten Featurevektoren.

Zur Transformation der Texte in einen Featurevektor wurden zwei verschiedene Large Language Models getestet: *Generative Pretrained Transformer 2* (GPT2) und das GPT3 Modell.

Für die Anomalieerkennung auf den extrahierten Vektoren wurden die pyOD Implementierung des Isolation Forest (IF), KNN und Autoencoders (AE) verwendet. PyOD [36] ist eine python-Bibliothek für die Erkennung von Outliern in multivariaten Daten. PyOD umfasst mehr als 40 Algorithmen zur Erkennung von Outliern und ist eine wichtige Ressource im akademischen, als auch kommerziellen Kontext.

In dieser Arbeit wurde bewusst nur minimales Hyperparameter-Tuning der genannten Algorithmen durchgeführt. Diese Entscheidung basiert auf der Zielsetzung, das Zusammenspiel der Algorithmen und der LLMs für verschiedene Probleme zu untersuchen und nicht die Architektur gezielt für eine spezifische Anwendung zu optimieren. Zudem ist es in realen Anwendungen nicht möglich Hyperparamter-Tuning durchzuführen, durch die größtenteils unsupervised Charakteristik der Anomalieerkennung.
Es wird untersucht, ob bestimmte Parameter Einfluss auf die Architektur haben, oder die Architektur robust gegen Anpassung der Parameter ist. Die Ergebnisse dieser Experimente sind in Kapitel 5.2.3 dargestellt.
Die Parameter wurden jeweils als Standardwerte der Algorithmen initialisiert.

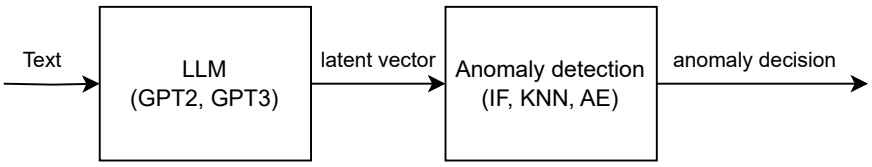

Abbildung 4.1: Die Architektur der Arbeit illustriert. Der Text wird durch den Encoderblock des LLM in eine Vektorrepräsentation transformiert. Der Vektor wird durch einen Anomalieerkennungsalgorithmus als Inlier oder Outlier klassifiziert.

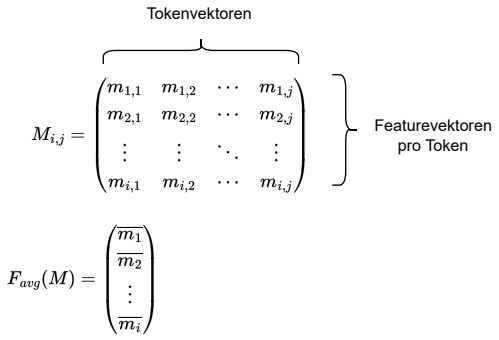

Abbildung 4.2: Aggregierungsfunktion F_{avg} illustriert

Die theoretische Grundlage der verschiedenen Algorithmen ist im Kapitel 2.4.1 definiert.

4.1 GPT2

Für die GPT2 Implemtierung wurde die *transformers* Bibliothek von Hugging Face [31] eingesetzt.
In der vorliegenden Arbeit wurde das Modell *gpt2* verwendet. Es gibt weitere Versionen des GPT2 Modells, wie *gpt2-large*, *gpt2-medium* und *gpt2-xl*, die sich in der Anzahl der Parameter unterscheiden. Die verschiedenen Versionen haben Parameterzahlen von 124 Millionen (*gpt2*) bis zu 1,5 Milliarden (*gpt2-xl*). Aufgrund begrenzter Ressourcen wurde in dieser Arbeit das kleinste Modell verwendet.
Für die Experimente wurden die Texte auf eine Länge von 100 Token begrenzt. Kürzere Texte werden mit sogenannten *padding tokens* aufgefüllt, denen im Attention Mechanismus kein Gewicht zugewiesen wird. Dadurch bleibt die Semantik des Satzes unverändert. Längere Texte werden nach 100 Token gekürzt.

Das Modell gibt für jeden einzelnen Token einen Vektorembedding der Länge 768 aus. Jeder Text ist damit gegeben durch eine Matrix $M \in \mathbb{R}^{768\times100}$. Die sehr hohe Dimension macht eine praktische Verarbeitung durch die Matrix nicht möglich. Aus diesem Grund wurden verschiedene Methoden verwendet, um die Dimension der Matrix zu reduzieren. Eine Möglichkeit ist die Funktion F_{avg}, die für die i-te Ausprägung den Durchschnitt über die j Tokenvektoren berechnet. (Abbildung 4.2)

$$F_{avg} : \mathbb{R}^{768\times100} \to \mathbb{R}^{768}$$
$$F_{avg}(M) = V$$

wobei für $i \in [1, 768]$, das i-te Element von V wie folgt berechnet wird:

$$V_i = \overline{m_i} = \frac{1}{100}\sum_{j=1}^{100} m_{i,j}$$

Im Kapitel 5 werden verschiedene Aggregierungfunktionen dargestellt und die Ergebnisse miteinander verglichen.

Die Wahl einer festen Tokenlänge von 100 für das GPT2 Modell wurde aus mehreren Gründen getroffen. Erstens erfordert die Reduktion der Texte mit dem Autoencoder die Verwendung einer festen Matrix, was bei variabler Länge nicht möglich wäre. Zweitens ist es das Ziel dieser Arbeit, eine Vielzahl von Experimenten durchzuführen. Ein Datensatz mit einer zu großen Anzahl von Tokens, wie beispielsweise der gdelt-Datensatz, hätte eine längere Verarbeitungszeit in Bezug auf die Rechenressourcen erfordert. Die Datensätze dieser Arbeit sind vergleichsweise klein. In realen Anwendungen können unsupervised Datensätze potentiell große Datenmengen sein und bestärken daher das Argument, die Tokenanzahl zu begrenzen.

4.2 GPT3

Als GPT3 Modell wurde die API von OPENAI [23] ausgewählt. Die API-Schnittstelle bietet die neusten und größten Modelle. Für alle Experimente wurde dabei die *text-embedding-ada-002* Variante verwendet. Dabei wird der Embedding eines gegebenen Textes ausgegeben und nicht die Ausgabe einer Aufgabe.

Im Gegensatz zum GPT2 Modell, war es für GPT3 nicht zwingend erforderlich die Tokenanzahl des GPT3-Modells zu begrenzen. Das Hauptziel der Arbeit besteht darin eine leistungsfähige Basisarchitektur zu entwickeln. Daher wurde sich in den Experimenten gegen eine Begrenzung der Tokenanzahl entschieden.
Im Gegensatz zu GPT2, bei dem für jeden Text Matrizen generiert werden, erzeugt GPT3 lediglich einen Vektor auf Textebene. Daher bestand keine Notwendigkeit, die Tokens zu begrenzen, weil die Transformation auf Textebene erfolgt und keine Matrizen erstellt werden müssen.
Darüber hinaus ermöglicht die API-Schnittstelle des GPT3-Modells eine effiziente Verarbeitung und Transformation der Texte, ohne dass eine Reduktion der Tokenanzahl erforderlich ist.

4.3 Anomalieerkennung

Für die Anomalieerkennung auf den extrahierten Vektoren wurde für die Experimente nur unsupervised Methoden verwendet, was auch der gängigen Methodik der Anomalieerkennung entspricht.
Im Bereich der Anomalieerkennung werden aus verschiedenen Gründen mehrheitlich unsupervised Methoden eingesetzt. Die Gründe sind, dass zum einen die Machine Learning-Pipeline von unsupervised Methoden kostengünstiger sind, da keine Labels im Vorhinein erstellt werden müssen. Zum anderen ist es in manchen Applikationen aufgrund der großen Datenmenge nicht möglich, vorher zu definieren, welche Ele-

mente “normal” bzw. “anomal” sind.
Eine weitere Herausforderung für die supervised Anomalieerkennung ist, dass in dem Trainingsdatensatz eine genügend hohe Anzahl an anomalen Datenpunkten vorhanden sein muss, damit der Algorithmus in der Lage ist, diese zu identifizieren. Außerdem müssen zu den verschiedensten Anomalien Repräsentanten im Datensatz vorhanden sein. Das wird erschwert, da sich die Anomalien im Laufe des Betriebs weiterentwickeln und immer neue Arten von Anomalien entstehen, die zu dem Zeitpunkt noch nicht im Trainingsdatensatz vorhanden waren.
Im Gegensatz zur supervised Learning, in dem das Modell auf Basis von gelabelten Trainingsdaten erstellt wird, werden unsupervised Methoden auf ungelabelten Daten trainiert, um Muster und Strukturen in den Daten zu identifizieren. Dabei wird die Annahme getroffen, dass der vorhandene Trainingsdatensatz nur “normale” (oder zu einem Großteil “normale”) Datenpunkte enthält. Anomalien werden dabei durch die Abweichung von den normalen Daten bestimmt. Das ermöglicht, dass nicht jede Anomalie im Vorhinein bekannt sein muss, sondern nur der normale Ist-Zustand.
Um das an einem Beispiel zu verdeutlichen: Es wird eine Raumsonde ins Weltall gesendet. Ein Anomalieerkennungssystem soll zur frühzeitigen Erkennung und Meldung potenzieller Fehler eingesetzt werden. Bei Verwendung eines supervised Ansatzes müssen sämtliche möglichen Fehler im Datensatz präsent sein und mit entsprechender Kennzeichnung versehen werden, um im Falle ihres Auftretens eine zuverlässige Erkennung zu gewährleisten.
Es ist leicht zu erkennen, dass die Berücksichtigung potenzieller Fehler im Voraus eine Herausforderung darstellt, da eine vollständige Bestimmung dieser Fehler äußerst schwierig ist.
Im Gegensatz dazu gestaltet sich die Modellierung des normalen Bereichs der Rakete als wesentlich einfacher. In realen Systemen genügt es außerdem häufig, das Vorliegen einer Anomalie zu erkennen, ohne eine Unterscheidung zwischen verschiedenen Arten anormalen Verhaltens vornehmen zu müssen.

5 Evaluation und Ergebnisse

5.1 Datensätze

5.1.1 Quellen

Für die vorliegende Arbeit wurden drei verschiedene Datensätze zur Validierung verwendet. Alle Datensätze sind öffentlich zugänglich. Als Daten wurden die nachfolgenden Quellen genutzt:

Der SMS Spam Datensatz [32] ist eine Sammlung von SMS Nachrichten aus verschiedenen Textkorpora, die zu einem Datensatz zusammengetragen wurden. Jede Nachricht, der Spalte *Text*, wird als spam oder ham gelabelt. Ham steht dabei für einen ormalen Datenpunkt, während ham eine Anomalie ist.

Als zweiten Datensatz wurde der "3 months of Nasdaq100 related news from GDELT" Datensatz von Kaggle verwendet [34]. Der Datensatz enthält Nachrichten von NASDAQ100 Unternehmen des GDELT Datensatzes. Die Nachrichten stammen aus einem Zeitraum vom 8. Juli 2022 bis zum 9. Oktober 2022. Die einzelnen Nachrichten, in der *body* Dimension, sind dem Thema entsprechend klassifiziert (Umwelt, Soziales, Governance). In der Arbeit wird die Kategorie Umwelt als Inlier gesehen und Soziales als Outlier.

Der dritte verwendete Datensatz ist der "New York Times Articles & Comments" von Kaggle. [3]
Für die Artikel wurden die Artikel verwendet, jedoch nicht die Kommentare. In der Datei *nyt-articles-2020.csv* befinden sich alle Zeitungsartikel der New York Times für den Zeitraum 1. Januar 2020 bis 31. Dezember 2020. In der Spalte *material* ist für jeden Artikel die zugehörige Art gegeben, die in der Arbeit als eine Entscheidung über Inlier oder Outlier benutzt wird. *News* sind dabei Inlier, während *Op-Ed* als Outlier betrachtet werden.

Im Kapitel 4 wurde beschrieben, dass die Vektoren der Texte auf eine Länge von 100 Tokens begrenzt wurden. Die deskriptiven Statistiken der Anzahl von Tokens der Originaldatensätze sind in Tabelle 5.1 gegeben. Es ist zu beobachten, dass größtenteils nur die Textdaten des GDELT Datensatzes gekürzt werden.

	SMS Spam	NYT	GDELT
mean	22.78	27.16	802.24
std	17.54	9.26	593.26
min	1	2	10
max	257	123	7106

Tabelle 5.1: Deskriptive Statistik für die Tokenanzahl der Texte. Die gegebenen Statistiken sind bezogen auf den GPT2 Tokenizer.

5.1.2 Tausch der Klassen

Die Zuweisung der Inlier- und Outlier-Klassen wurden umgekehrt, um die Auswirkung einer solchen Anpassung auf die Ergebnisse zu untersuchen. Die daraus ermittelten Ergebnisse wurden jedoch in der vorliegenden Arbeit nicht aufgeführt, da die zu geringe Datensatzgröße keine aussagekräftige Interpretation der Ergebnisse zuließ. Die Einschränkung der Datensatzgrößen kann zu ungenauen Ergebnissen und fehlerhaften Schlussfolgerungen führen, was die wissenschaftliche Aussage der Arbeit einschränkt. Daher wurden die Ergebnisse der Inversion der Inlier- und Outlier-Klassen in die vorliegende Arbeit nicht aufgenommen.

5.1.3 Binäre Datensätze

Die eingesetzten Datensätze sind, wie in Kapitel 5.1.1 beschrieben, Multiklassendatensätze. Da die Anomalieerkennung in binäre Klassen unterscheidet, müssen die Datensätze in einem ersten Schritt in ein binäres Datensatzformat transformiert werden. Hierbei wurde zwei Klassen ausgewählt und jeweils eine der Klassen als Inlier und die andere als Outlier deklariert. Die ausgewählte Klassen sind in Kapitel 5.1.1 beschrieben.
Für diese Arbeit wurde keine *one-vs-rest* Konstruktion gewählt. Die Multiklassendatensätze, die als Grundlage dienen, sind keine echten Anomalieerkennungsdatensätze, sondern Datensätze, in denen sich die verschiedenen Klassen potentiell ähnlich sein können. Bei der Verwendung einer *one-vs-rest* Konstruktion könnte Folgendes auftreten: Eine der beiden Klassen wird als normal deklariert, die andere Klasse, die als Ausreißer bezeichnet wird, kann jedoch eine gewisse Ähnlichkeit mit der normalen Klasse aufweisen. Eine Anomalie wird jedoch typischerweise als Punkt definiert, der sich deutlich von den normalen Punkten unterscheiden sollte und keine Nähe zu ihnen besitzt. Aus diesem Grund wurden zwei Klassen ausgewählt, die eine gewisse Distanz voneinander aufweisen sollten, statt einer *one-vs-rest* Konstruktion.

Eine weitere Einschränkung bei der Wahl war eine respektable Größe der ausgewählten Klasse für eine effektive Datensatzkonstruktion.

5.1.4 Train Test Split

Im Kontext der Evaluierung von Algorithmen auf binären Datensätzen werden diese üblicherweise in Trainings- und Testdatensätze aufgeteilt. Wie in Kapitel 4 beschrie-

Train with Outlier	Train/Test samples	GDELT	SMS	NYT
no	Train	(1090,0)	(3860,0)	(10470,0)
	Test	(218, 108)*	(965,747)*	(2618,2053)*
yes	Train	(872, 87)	(3860,386)	(10470,1047)
	Test	(218, 21)*	(965,361)*	(2618,1006)*

Tabelle 5.2: Größenverhältnisse von Trainings- und Testdatensätzen mit und ohne Ausreißer für GDELT, SMS und NYT Datensätze mit $(|I|, |O|)$ für I die Menge der Inlier und O die Menge der Outlier für den Fall. Bei allen Werten wurde *ratio_negatives* = 0.1 verwendet.

* Das Verhältnis von Inliern und Outliern ist nicht 50/50.

ben, wird für die vorliegenden Arbeit angenommen, dass der Großteil der Daten im Trainingsdatensatz normal ist, da es sich um ein Anomalieerkennungs-Setup handelt. Um die Auswirkung dieser Annahme auf die Ergebnisse zu untersuchen, werden zwei Fälle betrachtet:

- Mit Outliern im Trainingsdatensatz, wobei das Verhältnis mit Undersampling auf einen fixierten Wert von *ratio_negatives* = 0.1 festgelegt wird (*w Outlier*)
- Ohne Outlier im Trainingsdatensatz (*w/o Outlier*)

Der Paramater *ratio_negatives* wurde für die Experimente fixiert. Die Auswirkung durch Veränderung des Parameters auf die Ergebnisse wird in Kapitel 5.2.4 näher untersucht.

Für beide Fälle wurden Trainings- und Testdatensätze konstruiert. Die Anzahl der Outlier wurde im Testsatz auf 0.5 festgelegt wird, um eine gleichmäßige Verteilung der Klassen bei der Evaluation zu gewährleisten. Aufgrund der gegebenenfalls nicht idealen Verteilung, bezüglich der Anzahl an Inlier- und Outlier-Elementen, war eine exakte 50/50 Verteilung nicht immer möglich. Insbesondere für Trainingsdatensätze mit Outliern wurde die korrekte Anzahl an Outliern im Verhältnis *ratio_negatives* priorisiert. Die resultierenden Größen der drei Datensätze sind in der Tabelle 5.2 aufgeführt.

Die Zufallsauswahl der Trainings- und Testdatensätze erfolgte dabei pseudozufällig unter Verwendung eines festgesetzten Parameters, des sogenannten *random_state*. Dieser Parameter ermöglicht die Reproduzierbarkeit der Ergebnisse, indem er sicherstellt, dass die Zufallszahlen bei jeder Ausführung des Algorithmus aus demselben Seed generiert werden.
Des Weiteren wurden in den Experimenten verschiedene Werte des *random_states* verwendet, um die Wahrscheinlichkeit zufälliger Ergebnisse zu minimieren.

5.2 Experimente

5.2.1 Gpt2 Aggregierungsfunktionen

Um geeignete Methoden zu identifizieren, die potentiell gute Ergebnisse liefern, wurde zunächst eine Beschränkung auf einen spezifischen Datensatz vorgenommen. Dabei wurde der SMS Datensatz ausgewählt, da dieser als ein typischer Anomalie-Datensatz für Texte gilt. Anschließend wurden die erzielten Ergebnisse mit anderen Datensätzen validiert.

Als erstes LLM wurde das GPT2 Modell mit dem SMS Datensatz getestet. Die einfache und kostenfreie Anwendung macht es möglich, zunächst verschiedene Methoden zu testen.
Im Kapitel 4 ist dargestellt, dass GPT2 eine hohe Anzahl an Dimensionen ($\mathbb{R}^{768\times100}$) für jeden Text ausgibt. Diese hohe Anzahl erschwert eine effektive Verarbeitung und es ist eine Dimensionsreduktion notwendig.
Gestartet wurde mit dem Durchschnitt, Minimum, Maximum und der Standardabweichung. Die Standardabweichung zeigte dabei eine bessere Leistung als die anderen Funktionen.
Zur Einschätzung, ob die einzelnen Komponenten der Standardabweichung, $\sigma = \sqrt{E(X^2) - (E(X))^2}$, zu der herausstechenden Leistung führen, wurde auch das Quadrat des Mittelwerts $E(X)^2$ verwendet.
Die Verwendung zeigte eine Verschlechterung der Ergebnisse im Vergleich zu der Leistung der Standardabweichung und des Mittelwertes.

Agg Fkt	mit/ohne Outlier	IF	KNN	AE
min	w/o	0,86	0,88	0,90
	w	0,79	0,78	0,85
max	w/o	0,91	0,91	0,90
	w	0,82	0,81	0,84
mean	w/o	0,89	0,88	0,89
	w	0,87	0,79	0,87
std	w/o	0,91	0,89	0,89
	w	0,88	0,80	0,86
mean^2	w/o	0,86	0,79	0,88
	w	0,82	0,71	0,84

Tabelle 5.3: AUC-Score von GPT2 auf dem SMS Datensatz mit dem Minimum, Maximum, Durchschnitt, Standardabweichung oder das Quadrat des Mittelwerts als Aggregierungsfunktion.

Eine Überlegung war es, statt einer Aggregierungsfunktion, einzelne Tokenvektoren auszuwählen. Die einzelnen Vektoren der Token erhalten auch Information des restlichen Textes im Encoder. Aus diesem Grund wurde getestet, wie gut ein einzelner Vektor den Text abbilden kann. Die einzelnen Dimensionen haben keine

bestimmte inhaltliche Interpretation, daher wurde exemplarisch der Token Vektor 0,50 und 99 ausgewählt. Der Vektor 50 erzielte dabei mit 0.85 die besten Ergebnisse (im Vergleich: 0.7 Tokenvektor 0, 0.72 Tokenvektor 99). Die Ergebnisse lassen darauf schließen, dass die mittleren Token mehr Inhalt enthalten, als äußere Vektoren. In Kapitel 5.2.7 wird für die Auswahl des Tokenvektors eine detailliertere Evaluation durchgeführt.

Im nächsten Schritt wurden die Komposition der drei Vektoren betrachtet (Konkatenation der drei Vektoren mit einer resultierenden Gesamtlänge von(2304,)). Die Komposition zeigte eine Verschlechterung des AUC Werts auf 0.8. Die Zusammenführung verschiedener Vektoren führt somit zu einer Verschlechterung gegenüber der Verwendung des mittleren Vektors.

Agg Fkt	mit/ohne Outlier	IF	KNN	AE
Vektor 0	w/o	0,71	0,72	0,68
	w	0,62	0,58	0,64
Vektor 49	w/o	0,85	0,85	0,84
	w	0,84	0,82	0,83
Vektor 99	w/o	0,72	0,71	0,73
	w	0,68	0,59	0,70
[V0, V49, V99]	w/o	0,78	0,83	0,80
	w	0,75	0,78	0,77

Tabelle 5.4: AUC-Score von GPT2 auf dem SMS Datensatz für die Auswahl des ersten, mittleren, letzten, oder alle drei als Kombinationsfunktion.

Alternativ zur Aggregierungsfunktion können Convolutional Autoencoders zur Dimensionsreduktions des Embeddings eingesetzt werden. Die anderen Aggregierungsfunktionen, die Werte in den Dimensionen über die Tokens zusammengefasst haben, erlaubten keine Selektion einzelner Werte. Der Autoencoder lernt eine latente Repräsentation, und ermöglicht es so die Dimensionen unterschiedlich zu gewichten. Für die Tests der vorliegenden Arbeit wurde eine 2500 dimensionale latente Repräsentation gewählt.
Hierbei wurde ein Convolutional Autoencoder eingesetzt, der die räumliche Struktur von Sätzen besser abbilden kann.
Um die Strukturen zwischen den Tokens zu identifizieren, wurden dazu eindimensionale Kernels entlang der Tokens verwendet. Da kein Zusammenhang zwischen zwei benachbarten Dimensionen in einem Token Vektor besteht, wurde sich gegen zweidimensionalen Kernels entschieden.
Nach dem Traingsprozess wurde der Encoder Block für die Reduktion der Dimensionalität verwendet. Anschließend wurden die latenten Vektoren zur Anomalieerkennung verwendet. In der Abbildung 5.1 ist zu erkennen, dass der Autoencoder in der Lage ist, die komplexen Strukturen der Matrix abzubilden.

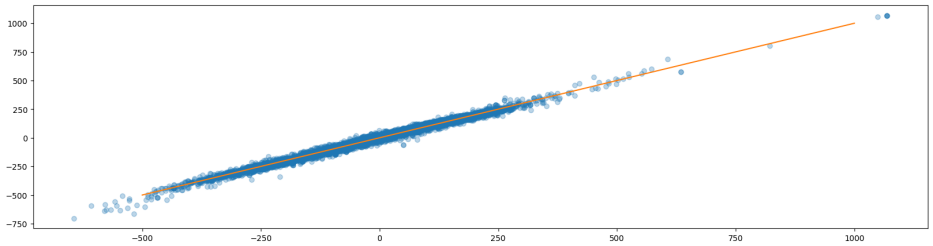

Abbildung 5.1: Erster Eigenvektor der PCA des Convolutional Autoencoder. Zusehen ist auf der x Achse der Input, auf der y Achse der Output des Autoencoders. Die orangene Linie symbolisiert eine perfekte Rekonstruktion.

Die Ergebnisse sind vielversprechend, mit dem Wissen, dass noch ein hohes Potential zur Optimierung vorhanden ist, die die Leistung potentiell verstärken kann.

Agg Fkt	mit/ohne Outlier	IF	KNN	AE
conv AE	w/o	0,84	0,81	0,90
	w	0,77	0,76	0,84

Tabelle 5.5: AUC-Score von GPT2 auf dem SMS Datensatz mit einem Convolutional Autoencoder als Kombinationsfunktion.

5.2.2 Vergleich GPT2/GPT3

In diesem Hauptexperiment wurde auf den Ergebnissen des Kapitels 5.2.1 aufgebaut, in denen sich die Standardabweichung als potentiell beste Aggregierungsfunktion herausgestellt hat. Die Leistung des GPT2 Modells wurde in diesem Experiment mit dem GPT3 Modell, als zweites LLM, verglichen. Die Evaluation wurde dabei erweitert auf die drei verschiedenen Datensätzen nyt, gdelt und SMS Spam. Die Übersicht der Ergebnisse zeigt Abbildung 5.2.

In den Ergebnissen des NYT-Datensatzes ist eine auffällige Charakteristik zu erkennen. Die AUC-Werte liegen entweder bei guten Werten nahe 0.7, oder bei unzureichenden Ergebnissen bei 0.5.

GPT2 zeigt auf dem Gdelt-Datensatz eine niedrige Leistungsfähigkeit, die Hinzufügung der Outlier steigerte die Leistung in diesem Fall marginal.
GPT3 hingegen erkennt klare Muster in den Daten. Eine mögliche Erklärung für diesen Unterschied ist, dass der eingesetzte Datensatz komplexer ist. Die Unterscheidung von Texten im Bereich Soziales und Umwelt nicht so einfach differenzierbar und benötigt semantisches Verständnis der Texte. Auch ein Mensch wird leichter zwischen spam und ham in den SMS Nachrichten differenzieren können, als das Thema eines Artikels zu erkennen. Für die Anomalieerkennung muss diese Differenzierung zusätzlich im Embedding abgebildet sein.

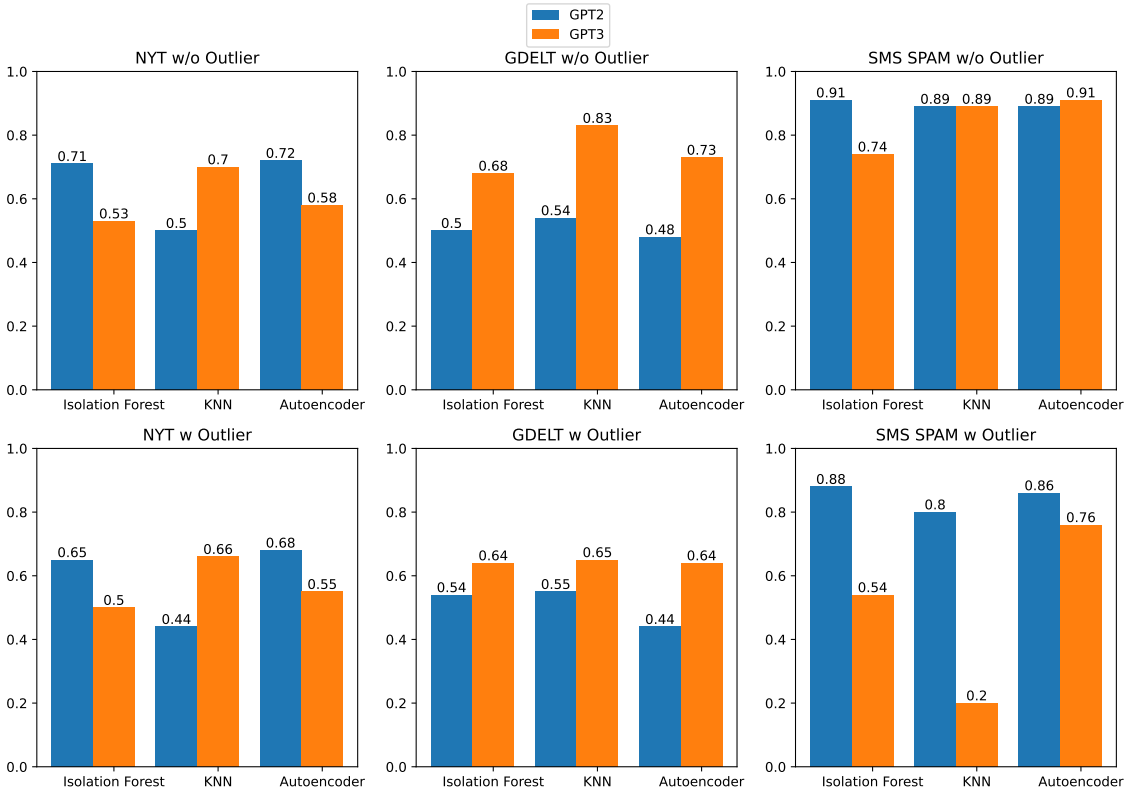

Abbildung 5.2: Leistung des Isolation Forest, KNN und Autoencoders auf den Embeddings des GPT2- (blau) und GPT3-Modells. In der Reihe unterschieden in dem Datensatz, in der Reihe durch das Hinzufügen von Outliern im Trainingsdatensatz (obere Reihe: ohne Outlier, untere Reihe: mit Outlier)

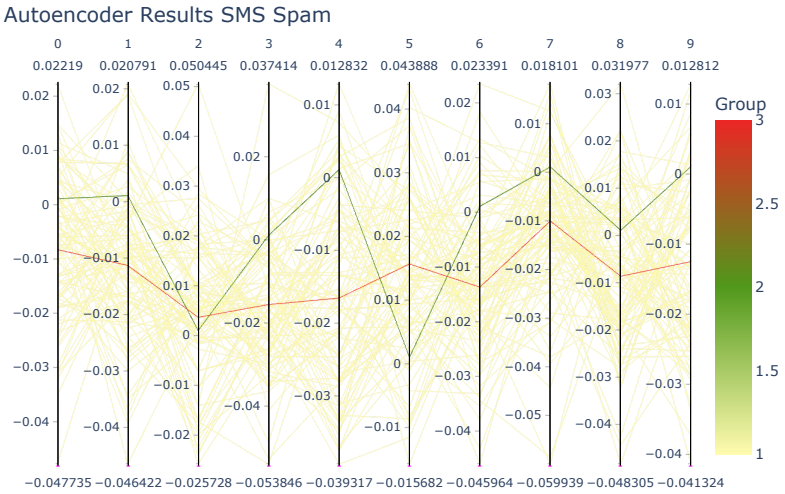

Abbildung 5.3: Parallelkoordinaten der ersten 10 Dimensionen der Embeddings des SMS Spam Datensatzes für den Evaluationsdatensatz (gelbe Linien) und dessen Rekonstruktion durch ein Autoencoder (grüne Linie). In rot gegeben, der Durchschnitt des Trainingsdatensatz.

Der SMS Spam Datensatz zeigte eine bessere Leistung bei Anwendung des GPT-2 Modells. Die Aggregierung mit der Standardabweichung könnte hier besonders gut funktionieren, da sie einzelne herausstehende Werte bzw Charakteristiken der Spam Nachrichten abbildet.

Eine allgemeine Beobachtung ist, dass eine deutliche Tendenz zur verbesserten Leistung des GPT3 in Verbindung mit dem KNN über die verschiedenen Experimente. Diese Beobachtung lässt sich möglicherweise durch die komplexe Struktur der Repräsentation erklären, die im GPT3 vorhanden ist. Der KNN, der wenige Annahmen über die zugrunde liegenden Daten trifft, scheint folglich in der Lage zu sein, diese komplexe Struktur effektiv zu modellieren und abzubilden.
Die einzige Abweichung der Tendenz zeigt der KNN auf dem SMS Spam Datensatz mit Outliern im Trainingsdatensatz. Dieser spezielle Testfall weicht jedoch von dem Aufbau eines typischen Anomalieerkennungsaufbau ab und kann daher vernachlässigt werden. Weitere detaillierte Informationen zu diesem Ergebnis sind im Kapitel 5.2.8 zu finden.

5.2.3 Robustheit gegenüber Parameterveränderung

In diesem Kapitel wird der Einfluss von Parameterveränderung auf die Leistung des Modells untersucht.
Die Robustheit soll dabei anhand des Einflusses des Parameters *hidden_neurons* auf

die Leistungsfähigkeit des Autoencoders untersucht werden. Die Experimente wurden auf den GPT3 Embeddings des SMS Spam Datensatz durchgeführt.
Die Ergebnisse zeigen übereinstimmende Genauigkeit bis zur vierten Nachkommastellen für verschiedene Netzwerkarchitekturen. Diese Beobachtung ist ungewöhnlich, da in der Regel eine zufällige Initialisierung der Gewichte Auswirkungen auf die Ergebnisse hat.

Ein Erklärungsansatz für diese Ergebnisse ist, dass der Autoencoder keine Repräsentation der Daten erlernt, sondern die Daten auf einen Punkt abbildet.

Die Vermutung wird durch den Vergleich der Standardabweichung des Evaluationsdatensatzes mit der Standardabweichung der Rekonstruktion bestätigt. Der Evaluationsdatensatz weist eine Standardabweichung von 0.026, im Gegensatz zu 0.00001 bei der Rekonstruktion auf. Der Wert nahe 0 suggeriert die Abbildung auf einen konstanten Punkt durch den Autoencoders.
Die Parallelkoordinaten in Abbildung 5.3 bestätigen anschaulich die Projektion auf einen konstanten Punkt. Die gelben Linien zeigen die Werte der 100 Datenpunkte des Evaluationsdatensatzes in den ersten 10 Dimensionen. Die grüne Linie zeigen für die 100 Datenpunkte den rekonstruierten Punkt. In der Darstellung sofort erkennbar, dass die Datenpunkte auf einen einzelnen Punkt abgebildet werden.

Auffällig ist bei diesen Ergebnis, dass der konstante Punkt vom Durchschnitt des Trainingsdatensatzes (rote Kurve) abweicht.

Die Abbildung auf den konstanten Punkt zeigt im Falle des SMS Datensatzes eine hohe Leistungsfähigkeit, die mit der Verteilung der Outliern und Inliern im Testdatensatz zusammenhängt.
Für eine alternative Verteilung könnte diese Abbildung eine niedrige Leistungsfähigkeit erzielen.

Der Autoencoder ist eine Blackbox und lässt keine Interpretierbarkeit der Ergebnisse im Vorhinein zu. Der Isolation Forest, als Machine Learning Algorithmus, besitzt im Gegensatz dazu eine Definition zur Detektierung von Anomalien.
In der Arbeit wird der Autoencoder der pyod Bibliothek verwendet. Die Anpassung zu einem lernfähigen selbsterstellten Autoencoder konnte aus Zeitgründen nicht in diese Arbeit mit aufgenommen werden.

5.2.4 Robustheit gegenüber Anzahl Outlier

In einem weiteren Experiment wurde die Robustheit der Algorithmen gegenüber der Anzahl Outlier im Trainingsdatensatz überprüft.

In den Experimenten der Arbeit wurde die Leistung der Algorithmen für einen Anomalieanteil im Datensatz von 0 (*w/o outlier*) und 0.1 (*w outlier*) untersucht.

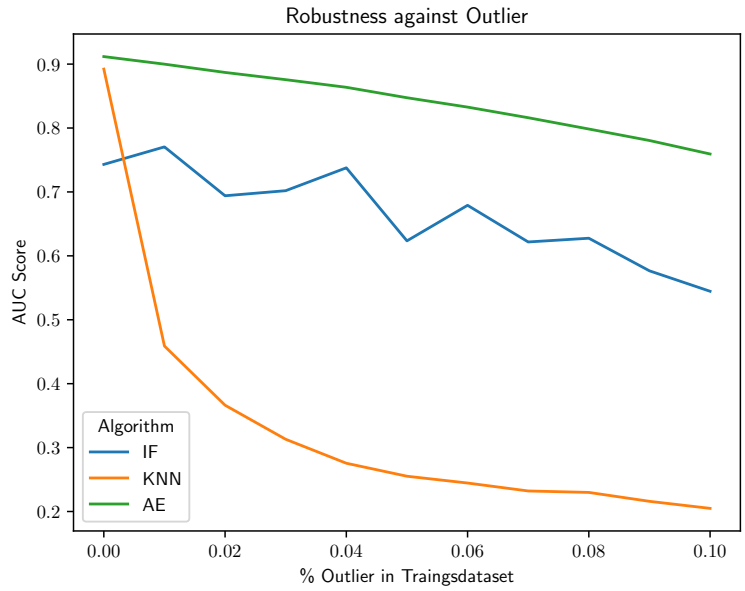

Abbildung 5.4: Veränderung der Leistung in Abhängigkeit von der Anzahl der Ausreißer im Trainingsdatensatz für verschiedene Algorithmen. Die Werte sind gegeben für GPT3 Embeddings des SMS Spam Datensatzes.

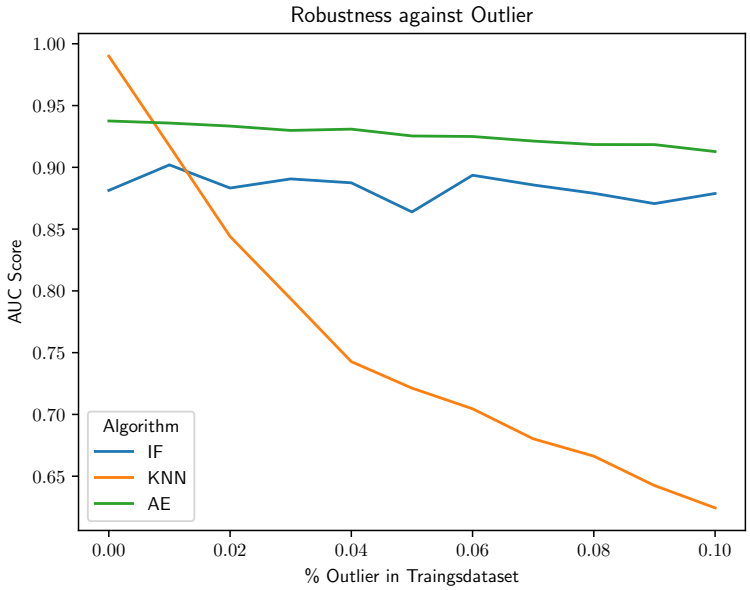

Abbildung 5.5: Veränderung der Leistung in Abhängigkeit von der Anzahl der Ausreißer im Trainingsdatensatz für verschiedene Algorithmen. Die Werte sind gegeben für GPT3 Embeddings des Enron Datensatzes.

In diesem Experiment (Abbildung 5.4) wird das Intervall zwischen den beiden Werten genauer betrachtet. Für die Evaluation wurde die GPT3 Embeddings des SMS Datensatz verwendet. Für jeden Anteil an Anomalien wird die Leistung des IF, KNN und AE ausgewertet. Die Werte sind gegeben für Anteile an Anomalien von 0 bis 0.1 in Schritten von 0.01.

Der KNN zeigte einen exponentiellen Zusammenhang zwischen der Leistung und der Anzahl Outliern. Dieser starke Zusammenhang lässt sich durch die hohe lokale Dichte der Ausreißer erklären (s. Kapitel 5.2.8) und widerspricht der Annahme, dass Inlier dichter sind als Outlier. (Kapitel 2.4.1)
Der Isolation Forest und der Autoencoder zeigen beide eine ähnliche Charakteristik mit einem linearen Zusammenhang und gleicher Steigung.

Der SMS Datensatz weist eine besondere Charakteristik auf, bei der Outlier dichter sind als Inlier. Aus diesem Grund wird in diesem Experiment zusätzlich die Robustheit des Enron Datensatz betrachtet (Abbildung 5.5).
Der KNN zeigt ebenfalls auf den ersten Blick eine vergleichbare Kurve. Allerdings unterscheidet sich die Y-Achsen Skalierung der beiden Grafiken. Die Kurve hat eine geringere Steigung als die des SMS Datensatzes, zeigt aber weiterhin ein sensitives Verhalten, besonders im Vergleich zu dem Autoencoder und KNN.

Der Zusammenhang zwischen der Leistungsfähigkeit des KNNs und der Anzahl der Outliern hängt daher von der Dichte der Outlierklasse im Verhältnis zur Inlierklasse ab.
Der Isolation Forest und der Autoencoder zeigten ebenfalls eine ähnliche Charakteristik mit robustem Verhalten gegenüber der Anzahl der Outliern.

Die Werte des Isolation Forest folgen der gängigen Annahme, dass die Leistung für einen kleinen Anomalieanteil (hier 0.01) höher ist.

Das Experiment zeigt, dass der Isolation Forest und Autoencoder robust gegen die Anzahl an Outliern ist, während der KNN sensitiv zu der Anzahl an Outliern im Trainingsdatensatz ist.

5.2.5 Outlier Exposure

Im Rahmen der Bachelorarbeit wurde auch die Möglichkeit der Outlier Exposure in Kombination mit den GPT3 Embeddings untersucht.
Outlier Exposure unterscheidet sich von den anderen dargestellten Methoden. Bislang wurde von einem unsupervised Rahmen ausgegangen, bei dem der Trainingsdatensatz größtenteils Inlier enthält (Im Rahmen der Arbeit wird die Sensitivität zu der Anzahl Outlier im Trainingsdatensatz überprüft. s. Kapitel 5.2.4). Der zusätzliche Hilfsdatensatz ermöglicht es, die Anomalieerkennung in einem *unsupervised OE* [25] (in anderen Arbeiten oft auch als *semi-supervised* bezeichnet) Rahmen durchzuführen.

Für die Implementierung wurde der HSC Algorithmus aus dem GitHub Repository des Papers [25] verwendet und das Netzwerk in geeignete Dimensionen für die Verarbeitung mit einem LLM angepasst. Die Verwendung eines Autoencoders für die Bestimmung der initialen Gewichte wurde hierbei unterlassen.
Für die Outlier Exposure wurden zwei verschiedene Hilfsdatensätze ausgewählt.

- *Wikipedia Movie Plots* [14] Der Datensatz stellte eine Sammlung von 35000 Handlungen und Metadaten zu dem Film bereit.
- *Coronavirus tweets NLP* [20] Der Datensatz enthält eine Sammlung von Tweets zum Coronavirus.

Die Datensätze wurden ohne Vorgabe eines bestimmten Musters oder spezifischer Kriterien willkürlich ausgewählt. Diese Vorgehensweise wurde bewusst gewählt, um sicherzustellen, dass das entwickelte System eine breite Palette von Szenarien abdeckt und keine zusätzlichen wissenschaftlichen Vorannahmen bei der Datenauswahl erforderlich sind.
Es wurden zwei Datensätze verwendet, um den Einfluss der Datensätze auf die Leistung des Algorithmus zu überprüfen.

HSC wurde zuerst auf dem Wikipedia Datensatz angewendet. Die Evaluierung

Hilfsdatensatz	Datensatz	AUC Score
Wikipedia Plots	SMS	0,56
Wikipedia Plots	NYT	0,5
Corona NLP	SMS	0,73
Corona NLP	NYT	0,51

Tabelle 5.6: Outlier Exposure Ergebnisse des HSC Algorithmus für verschiedene Datensätze und Hilfsdatensätze

des Algorithmus ergab eine begrenzte Vorhersagekraft mit einem AUC Score von 0.56, insbesondere im Vergleich zu den Ergebnissen auf dem SMS-Spam-Datensatz. Im Anschluss daran wurde der Wikipedia-Hilfsdatensatz mit dem nyt-Datensatz angewendet, was zu einem nicht aussagekräftigen Ergebnis von 0.5 führte.

Um festzustellen, ob die Leistung des Algorithmus von der Auswahl des Hilfsdatensatzes abhängt, wurde ein zweiter Hilfsdatensatz herangezogen. Bei Anwendung mit des SMS-Datensatzes konnte eine deutliche Steigerung des AUC Score auf 0.73 festgestellt werden.

Des Weiteren wurde untersucht, ob der Corona-Datensatz im Allgemeinen eine bessere Leistung erzielt als der Wikipedia-Datensatz. Hierzu wurde der Corona-Datensatz ebenfalls in Kombination mit dem nyt-Datensatz trainiert. Es zeigte jedoch keine Verbesserung der Vorhersagekraft des Algorithmus.

Die Experimente zeigen, dass die Ergebnisse sensitiv gegenüber der Auswahl der

Hilfsdatensätze sind. Dies lässt sich damit erklären, dass bei einer schlechten Performance die Outlier näher an den Inliern des Datensatzes liegen als an den Werten des Hilfsdatensatzes. Dadurch werden die Outlier nahe an Inliern projiziert und erhalten niedrige Werte, ähnlich wie die Inlier. Dies lässt darauf schließen, dass die Leistung des Algorithmus sensitiv gegenüber der Auswahl der Hilfsdatensätze ist.

Die Arbeit von Matthiesen et al. [19] zeigt eine ähnliche Charakteristik. Das Experiment verwendet den CIFAR-Datensatz, der aus 32x32 Pixel Bildern verschiedener Klassen besteht. Das Experiment zeigt, dass das Hinzufügen von Bildern von Wölfen im Hilfsdatensatz einen positiven Effekt auf das Training hat, wenn die Anomalieklasse Hunde sind. Im Gegensatz dazu hat das Hinzufügen von Bildern von Walen in drei von vier Fällen einen negativen Effekt.
Die Arbeit zeigt, dass diese Ergänzung sowohl positive als auch negative Effekte haben kann.

Der positive Effekt des Hinzufügen von Wolfsbildern lässt sich darauf zurückführen, dass der erweiterte Hilfsdatensatz durch das Hinzufügen der Wolfsbilder eine nähere Repräsentation der Outlier (Hunde) ermöglicht.

Die Quellenrecherche für diese Arbeit ergab, dass bislang keine LLMs für die Transformation der Texte in Kombination mit Outlier Exposure veröffentlicht wurde. Diese neuartige Kombination könnte die bisher eher unzureichenden Ergebnisse erklären.
Obwohl Outlier Exposure noch weitere Anpassung erfordert, um in diesem Kontext zu funktionieren, ist es ein spannendes Thema mit Potenzial. Die Möglichkeit, ein unsupervised Training in ein semi-supervised Training umzuwandeln, eröffnet neue Möglichkeiten und ähnelt dem Ansatz der LLMs. Diese Modelle können durch das self-supervised Learning auf umfangreichen Datenmengen trainiert werden und sind in der Lage, komplexe Strukturen zu identifizieren. In einer Zeit, in der nahezu unbegrenzte Rechenleistung verfügbar ist, kann Outlier Exposure als semi-supervised Ansatz ähnliche Leistungssprünge ermöglichen.

5.2.6 Sentence Level

Eine weitere Methodik, die für diese Arbeit getestet wurde, war die Anwendung des GPT3 Modells auf Sentence-Level. Dabei wurden die Texte in einzelne Sätze unterteilt, für die jeweils ein Embedding erstellt wird. Anschließend wurde die Menge von Vektoren in geeigneter Weise zu einem Textvektor kombiniert. Für die Unterteilung der Texte in die einzelnen Sätze wurde eine Funktion aus Stackoverflow [8] verwendet. Die Transformation der Sätze in die Vektorrepräsentation wurde mit GPT3 erzielt. Zur geeigneten Kombination wurde der Mittelwert über die Dimensionen der Satzvektoren gebildet.
Die Verarbeitung der Texte auf Sentence Level zeigte auf dem SMS Spam Datensatz keine Leistungsfähigkeit. Auf dem Gdelt Datensatz erhöhte sich die Leistungsfähigkeit, aber zeigte dabei keine Verbesserung gegenüber der Verarbeitung

auf Text Level. Die zusätzliche Komplexität führte daher nicht zu einer Verbesserung und wurde aus diesem Grund nicht weiter in dieser Arbeit verfolgt.

Datensatz	Trainieren mit/ohne Outlier	IF	KNN	AE
SMS Spam	ohne	0,48	0,48	0,48
	mit	0,48	0,47	0,48
Gdelt	ohne	0,66	0,81	0,64
	mit	0,63	0,68	0,61

Tabelle 5.7: AUC-Score des GPT3 auf Sentence-Level und anschließender Mittelwertaggregierung. Gegeben sind die Daten für die zwei Datensätze SMS Spam und Gdelt

5.2.7 Tokenvektor Aggregierungsfunktion

Im Kapitel 5.2.1 wurde die Tokenvektor-Selektion als Aggregierungsfunktion eingeführt, die in diesem Abschnitt detaillierter betrachtet wird. Für dieses Experiment wurde nicht nur exemplarisch der erste, mittlere und letzte Vektor (0,49,99) ausgewählt (s. Kapitel 5.2.1), sondern jeder Vektor. Die Leistung wird bemessen durch den AUC-Score in Kombination eines KNN und sind in Abbildung 5.6 illustriert.
Die Ergebnisse auf dem SMS Spam Datensatz zeigten vergleichbare Werte für die drei Anomalieerkennungsalgorithmen. Dieses Experiment hatte nicht zum Ziel die potentiell besten Ergebnisse bei der Auswahl eines Tokenvektors zu finden, sondern einen möglichen Zusammenhang zwischen der Auswahl eines Vektors und der resultierenden Leistung des Modells. Aus diesem Grund wurden nur die Ergebnisse des KNN betrachtet.
Das Experiment wurde erweitert auf dem NYT und GDELT Datensatz.

Die Kurve des SMS Datensatzes bestätigt die Vermutung aus Kapitel 5.2.1, dass der mittlere Vektor auf dem SMS Datensatz zu einem besseren Ergebnis tendiert, als der erste und letzte Vektor. Das Maximum ist jedoch nicht bei Vektor 50, sondern bei 28.

Der Gdelt Datensatz zeigt keine Korrelation zwischen ausgewähltem Tokenvektor und Leistung. Im Gegensatz zu dem SMS Datensatz, ist die durchschnittliche Anzahl Tokens des Gdelt Datensatz (802) größer als die maximale Token Länge in der Arbeit (100). Aus diesem Grund sind die Ergebnisse nicht sensitiv zur Tokenauswahl.

Der NYT Datensatz zeigt eine ähnliche Charakteristik zu dem SMS Datensatz. Die Kurve ist jedoch gespiegelt, mit dem Minimum an Index Tokenvektor 20.

Eine mögliche Erklärung für die steigende Leistung des SMS Datensatz, ist die unterschiedliche Länge der Inlier- und Outlier-Ausprägungen.
Spam Texte sind im Durchschnitt, mit 41.68 länger als Ham Texte mit 19.86.

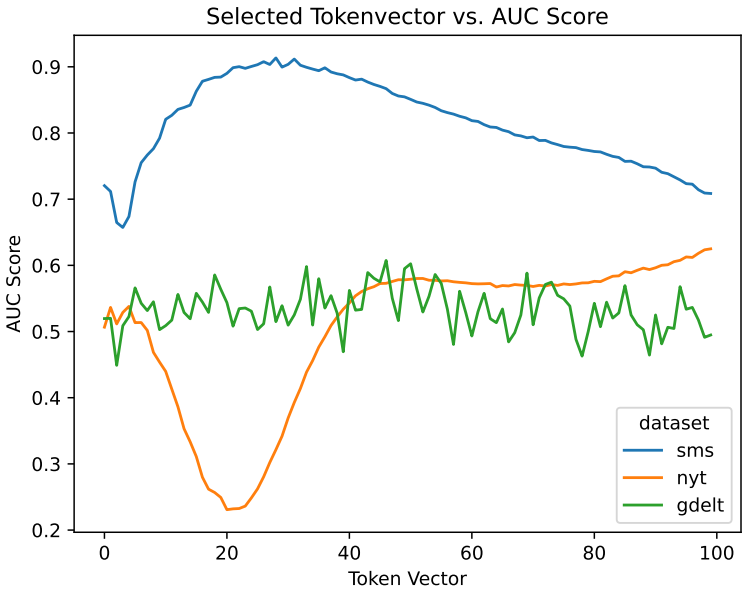

Abbildung 5.6: Auswahl verschiedener Tokenvektoren als Gpt2 Aggregierungsfunktion gegen resultierenden AUC Wert visualisiert. Die Werte sind für die 3 Datensätze des SMS Spam, NYT und Gdelt Datensatzes gegeben.

Der Abbildung 5.7 ist zu entnehmen, dass die Embeddings der Padding Tokens dichter als die der Wörter sind. Ein Inlier Wort, hat, durch die kürzere durchschnittliche Anzahl Tokens, wahrscheinlicher ein Padding Token an Stelle 28. Die Padding Tokens an Index 28 besitzen eine höhere Dichte. Daraus folgt, dass Inlier wahrscheinlicher als Inlier klassifiziert werden. Die Kurve steigt also an bis zum Index 28, denn die Wahrscheinlichkeit, dass die Inlier Texte Padding Tokens an der Stelle enthalten, die dichter sind und dementsprechend als Inlier klassifiziert werden, größer wird. Für Indexe größer als 28 steigt die Zahl der Outlier mit Padding Token überproportional zu der Anzahl Inlier mit Padding Token und die Leistung nimmt ab.

Der NYT Datensatz sind die Längenverhältnisse zwischen Inlier und Outlier invers, welches auch dem Graphen zu entnehmen ist. Die Outlier Texte sind kürzer als die Inlier Texte und dementsprechend sinkt der AUC Score mit zunehmenden Index.

Daraus kann gefolgert werden, dass die Leistung der Anomalieerkennung sensitiv zur Auswahl des Tokenvektors ist, wenn die Texte kürzer sind als die maximale Anzahl der Token. Die Leistung ist, im Fall kürzerer Texte, durch die Länge der Texte der Inlier und Outlier Klassen bestimmt.

Die im Rahmen des Experiments gewonnene Beobachtung bietet zusätzliche Erkenntnisse über die Leistung des GPT2 Modells auf dem SMS Datensatz.
Die Standardabweichung wird über die 100 Tokens gebildet. Die Padding Tokens unterscheiden sich von den restlichen Tokens. Die Standardabweichung, die als Ag-

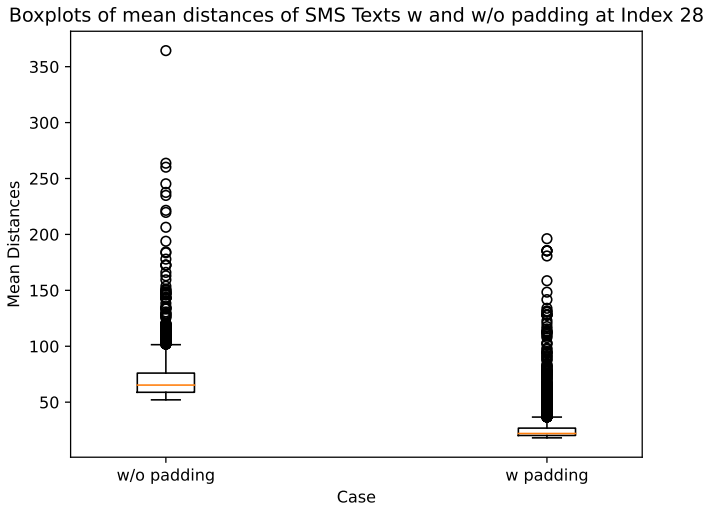

Abbildung 5.7: Mittlere Distanz zwischen den Elementen der beiden Klassen: Texte, die an Index 28 kein Padding Token enthalten (w/o padding) und einmal Texte, die an Index 28 ein Padding Token enthalten.(w padding) Die Werte beziehen sich auf GPT2 Transformierung des SMS Datensatzes.

gregierungsfunktion verwendet wurde, steigt mit zunehmender Anzahl an Padding Tokens. Inlier, die im Durchschnitt kürzer sind, unterscheiden sich von den Outliern durch eine höhere Standardabweichung.
Diese Annahme bestätigt der AUC Score von 0.89 eines einfachen Klassifizierers, der für die Bestimmung der Anomalien, nur die Länge der Textdaten verwendet.

5.2.8 Leistungseinbrüche KNN

Im Rahmen des SMS Datensatzes zeigte der KNN ein herausstechendes Ergebnis für den Fall des Trainings mit Outliern. Der Algorithmus erzielte einen AUC Score von 0.2 (Abbildung 5.2). Die Klassifizierung der Klassen ist im Fall von AUC Werten kleiner 0.5 invertiert, Inlier werden als Outlier und Outlier als Inlier klassifiziert.
Der KNN verwendet für die Klassifizierung in Inliern und Outlier das Prinzip der lokalen Dichte. Für einen AUC Wert kleiner als 0.5 liegt es nahe, dass die Outlier Klasse dichter als die Inlier Klasse ist. Für die Berechnung der Dichte der beiden Klassen wurde für jeden Datenpunkt die mittlere Distanz zu allen anderen Datenpunkten der Klasse berechnet. Die Ergebnisse sind in der Abbildung 5.8 dargestellt. Die Illustration zeigt deutlich, dass die Dichte der Inlier Klasse niedriger ist, als die der Outlier und bestätigt damit die vorhergehende Vermutung.

Die Leistung unterhalb 0.5 für den KNN lässt darauf schließen, dass die Annahme in realen System, Inlier hätten eine höhere Dichte, nicht in jedem Fall gilt.

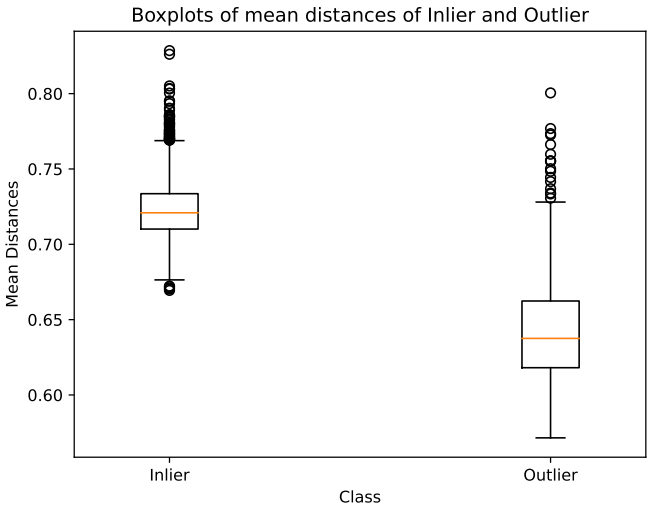

Abbildung 5.8: Für die beiden Klassen des SMS Spam Datensatzes jeweils ein Boxplot, der die mittlere Distanz zwischen den Elementen der Klassen abbildet. Auf der linken Seite die Inlier Klasse (ham) und auf der rechten Seite die Outlier Klasse (spam)

5.2.9 Enron Datensatz Leistungsfähigkeit GPT3

Zum Abschluss der Arbeit wurden zusätzliche Überprüfungen auf dem Enron-Datensatz durchgeführt. Aufgrund der Bearbeitung des Datensatzes zum Abschluss des Arbeit war dieser Datensatz nicht Teil der Hauptexperimente. Dennoch wurde es als wichtig erachtet, den Datensatz mit in diese Arbeit einzubeziehen, um die Ergebnisse auf den übrigen Datensätze zu validieren.

Die durchgeführten Überprüfungen (Abbildung 5.9) zeigen eine klare Dominanz von GPT3 gegenüber dem GPT2-Modell, wobei insbesondere die Leistung in Verbindung mit dem KNN hervorstechen, wie bereits in Kapitel 5.2.2 herausgestellt wurde. Die Leistung des KNN in Verbindung mit GPT3 auf dem Datensatz ohne Ausreißer ergibt mit einem Wert von 0.99 ein nahezu perfektes Ergebnis.

Diese Ergebnisse unterstreichen die Überlegenheit von GPT3 und bestätigen die Vorteile bei der Verarbeitung von Texten, insbesondere in Kombination mit dem KNN. Die Einbindung des Enron-Datensatzes ermöglichte eine zusätzliche Validierung und unterstützt die Schlussfolgerungen, die aus den Hauptexperimenten gezogen wurden.

5.2.10 Differenzierung von Gefühlen

Textdaten in natürlicher Sprache unterscheiden sich nicht nur auf der inhaltlichen Ebene, sondern auch auf der emotionalen Ebene.
Die Differenzierung von Textdaten auf der inhaltlichen Ebene wurde in dieser Arbeit bereits durch die vier Datensätze behandelt.

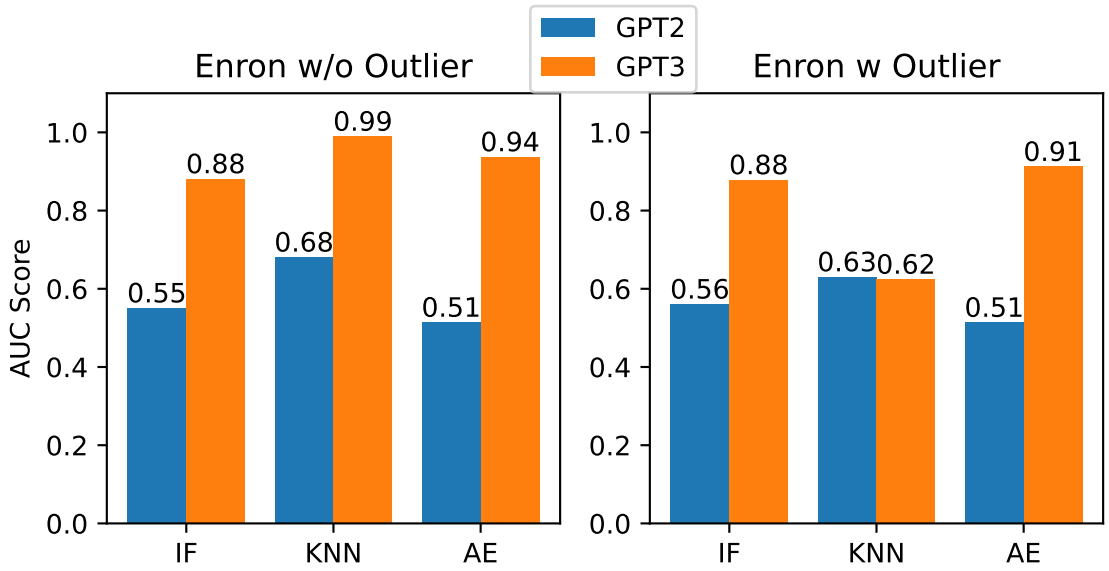

Abbildung 5.9: AUC Score des Enron Datensatzes für die verschiedenen Algorithmen und LLMs

Angesichts des wachsenden Interesse an der automatisierten Verarbeitung von Textdaten, ist es von großem Interesse, dass unsupervised Anomalieerkennungsalgorithmen die Fähigkeit entwickeln, menschliche Gefühle zu differenzieren.

Um zunächst festzustellen, ob LLMs in der Lage sind, diese Emotionen unterschiedlich abzubilden und ob Anomalieerkennungsalgorithmen in der Lage sind, diese voneinander zu trennen, wurde in dieser Arbeit der Datensatz *Emotion Detection from Text* [10] verwendet.

Die Ergebnisse (Abbildung 5.10) zeigen ausschließlich die Leistung für GPT3. GPT2 zeigte bei den vorangegangenen Experimenten bereits schlechtere Ergebnisse, daher wurden vergleichende Experimente zwischen GPT2 und GPT3 nicht durchgeführt.
Die *happiness* Klasse wurde dabei als Inlier und *sadness* als Outlier festgelegt.
Dieses Experiment zeigt, dass das Modell für komplementäre Gefühle (*happiness* und *sadness*) eine gewisse Separierung darstellen kann.
Weiterführende Experimente könnten diese wichtige Unterscheidung zwischen Gefühlen weiter untersuchen und überprüfen, in wieweit nah verwandte Emotionskategorien, wie beispielsweise Begeisterung und Freude, sich unterscheiden lassen können.
Eine weitere spannende Frage im Bezug zu der Separierung der Gefühle ist, ob einzelne Komponenten des Vektors die Gefühlslage abbilden und damit für eine Separierung genügen, oder ob die Gesamtheit des Vektors das spezifische Gefühl repräsentiert.

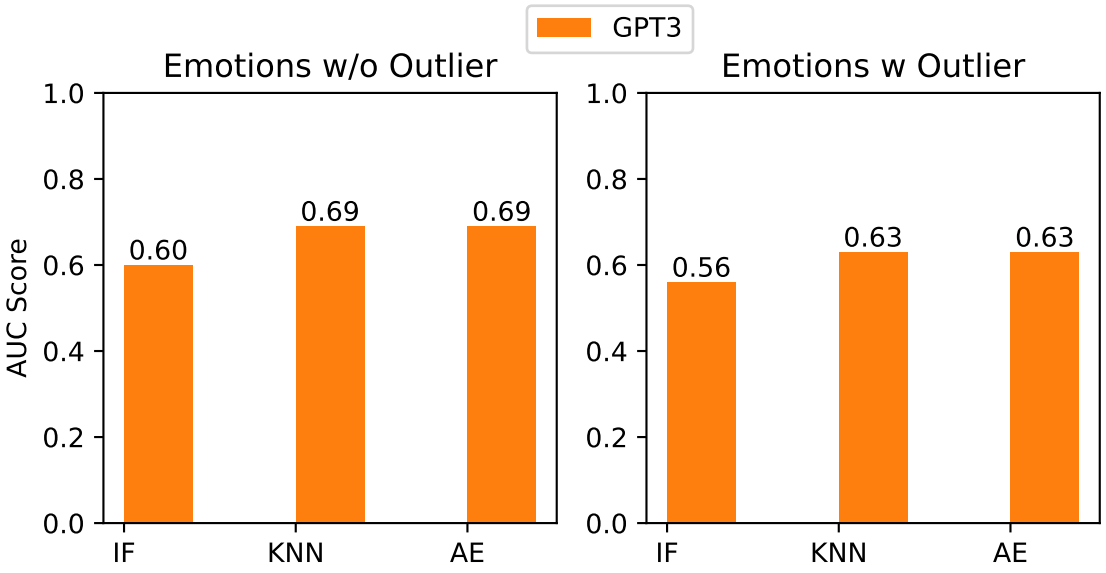

Abbildung 5.10: AUC Score des Tweet Emotions Datensates auf GPT3 für den IF, KNN und Autoencoder

6 Fazit

Die vorliegende Arbeit geht der Frage nach:
“Wie wirkt sich die Kombination verschieder LLMs und Anomalieerkennungsalgorithmen auf die Leistungsfähigkeit der Anomalieerkennung aus?”

Für die Entwicklung einer geeigneten Basisarchitektur wurden verschiedene LLMs mit unterschiedlichen Algorithmen verglichen. Zusätzliche Experimente lieferten ergänzende spezifische Einblicke in die Leistungsfähigkeit der Algorithmen.

Die zu Beginn der Tests angenommene hohe Leistungsfähigkeit des verwendeten GPT2-Modells beschränkt sich ausschließlich auf den verwendeten SMS Datensatz.
Die vergleichenden Experimente zwischen dem GPT2- und GPT3 Modell ergaben eine höhere generelle Leistungsfähigkeit des GPT3-Modells.
Dieses Ergebnis konnte durch Hinzufügens eines weiteren Datensatzes bestätigt werden.
Der KNN Algorithmus zeigt über die Experimente hinweg die höchste Leistungsfähigkeit gegenüber dem Autoencoder und dem Isolation Forest.
Die Experimente ergaben, dass der KNN sensitiv zu der Anzahl an Outliern im Traingsdatensatz ist, insbesondere im Vergleich zu dem Autoencoder und Isolation Forest. Der Grad der Sensitivität hängt mit dem Verhältnis der Dichte von Inliern zu Outliern zusammen.

Der Autoencoder zeigte hohe Ergebnisse. Ein zusätzliches Experiment zeigte jedoch, dass der Autoencoder die Daten ausschließlich auf einen Punkt abbildet. Die hohen Ergebnisse sind zufällig und hängen mit der Verteilung der Inlier- zu der Outlierklasse zusammen.

Ist eine geringe Anzahl an Outlier im Trainingsdatensatz bekannt, wird der KNN tendenziell bessere Ergebnisse darstellen. Bei einer bekannten hohen Anzahl überzeugt die robuste Architektur des Isolation Forest. Die Leistungsfähigkeit des Autoencoders kann den Nachteil der Blackbox-Charakteristik nicht kompensieren. Falls keine Informationen zu der Anzahl an Outliern vorliegt, ist der Isolation Forest die sichere Komponente.

Mit der Arbeit konnte eine leistungsfähige Basisarchitektur zur Anomalieerkennung auf Textdaten entwickelt werden. Diese setzt sich zusammen aus dem GPT3 Modell und den Anomalieerkennungsalgorithmen KNN oder Isolation Forest. Der geeignete Algorithmus ist abhängig von der Anzahl an Outliern im Trainingsdatensatz.

In der Arbeit wurde ein Ansatz aus den verwandten Arbeiten aufgegriffen. Die Verwendung des Outlier Exposure zeigte jedoch keine Verbesserung gegenüber den herkömmlichen unsupervised Methoden.
Der Ansatz hat das Potential, aufgrund der semi-supervised Eigenschaft, die Anomalieerkennung auf Textdaten deutlich zu verbessern. Dazu bedarf es weitere Modifikationen, die in zukünftigen Arbeiten entwickelt werden könnten.
Des weiteren könnten weitere Arbeiten GPT4 verwenden und die Vergleiche der der Leistungsfähigkeit der verschiedenen LLMs auf GPT4 erweitern.

Literaturverzeichnis

[1] Charu C. Aggarwal und Saket Sathe. Theoretical foundations and algorithms for outlier ensembles. *SIGKDD Explor. Newsl.*, 17(1):24–47, 2015. ISSN 1931-0145. doi: 10.1145/2830544.2830549.

[2] Kumar Ajitesh. Roc curve & auc explained with python examples, 2020. URL `https://vitalflux.com/roc-curve-auc-python-false-positive-true-positive-rate/`.

[3] Benjamin Dornel. New york times articles & comments, 2021. URL `https://www.kaggle.com/datasets/benjaminawd/new-york-times-articles-comments-2020`.

[4] David M. Blei, Andrew Y. Ng, und Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, und Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, und H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc, 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`.

[6] Dan Hendrycks, Mantas Mazeika, und Thomas Dietterich. Deep anomaly detection with outlier exposure, 2019.

[7] Markus Goldstein und Andreas Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: poster and demo track*, 1:59–63, 2012.

[8] D. Greenberg. Stackoverflow: How can i split a text into sentences?, 2015. URL `https://stackoverflow.com/a/31505798/14191859`.

[9] Xiaoyi Gu, Leman Akoglu, und Alessandro Rinaldo. Statistical analysis of nearest neighbor methods for anomaly detection. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc, Red Hook, NY, USA, 2019.

[10] Pashupati Gupta. Emotion detection from text, 2021. URL `https://www.kaggle.com/datasets/pashupatigupta/emotion-detection-from-text`.

[11] James A. Hanley und Barbara J. McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[12] Amir Jafari. A deep learning anomaly detection method in textual data, 2022.

[13] Jun S. Liu. The collapsed gibbs sampler in bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association*, 89(427):958–966, 1994. ISSN 01621459. URL `http://www.jstor.org/stable/2290921`.

[14] JustinR. Wikipedia movie plots, 2018. URL `https://www.kaggle.com/datasets/jrobischon/wikipedia-movie-plots`.

[15] Zheng Li, Yue Zhao, Nicola Botta, Cezar Ionescu, und Xiyang Hu. Copod: Copula-based outlier detection. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1118–1123, 2020. doi: 10.1109/ICDM50108.2020.00135.

[16] Fei Tony Liu, Kai Ming Ting, und Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008. doi: 10.1109/ICDM.2008.17.

[17] Amogh Mahapatra, Nisheeth Srivastava, und Jaideep Srivastava. Contextual anomaly detection in text data. *Algorithms*, 5(4):469–489, 2012. ISSN 1999-4893. doi: 10.3390/a5040469. URL `https://www.mdpi.com/1999-4893/5/4/469`.

[18] Mahsa Mohaghegh und Amantay Abdurakhmanov. Anomaly detection in text data sets using character-level representation. *Journal of Physics: Conference Series*, 1880(1):012028, 2021. doi: 10.1088/1742-6596/1880/1/012028.

[19] Jennifer Jorina Matthiesen und Ulf Brefeld. When more is less adverse effects in outlier exposure, 2022. URL `https://ml3.leuphana.de/publications/nldl2022a.pdf`.

[20] Aman Miglani. Coronavirus tweets nlp - text classification, 2020. URL `https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification`.

[21] Michael Nielsen. Using neural nets to recognize handwritten digits, 2019.

[22] Nils Reimers und Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[23] OpenAI. Openai api, 2022. URL `https://api.openai.com/v1/embeddings`.

[24] Tomáš Pevný. Loda: Lightweight on-line detector of anomalies. *Mach. Learn.*, 102(2):275–304, 2016. ISSN 0885-6125. doi: 10.1007/s10994-015-5521-0.

[25] Philipp Liznerski, Lukas Ruff, Robert A. Vandermeulen, Billy Joe Franks, Klaus-Robert Müller, und Marius Kloft. Exposing outlier exposure: What can be learned from few, one, and zero outlier images. *Transactions on Machine Learning Research*, 2022. URL `https://openreview.net/forum?id=3v78awEzyB`.

[26] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, und Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[27] David E. Rumelhart, Geoffrey E. Hinton, und Ronald J. Williams. Learning internal representations by error propagation. In David E. Rumelhart und James L. Mcclelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, pages 318–362. MIT Press, Cambridge, MA, 1986.

[28] Mayu Sakurada und Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, MLSDA'14, pages 4–11, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450331593. doi: 10.1145/2689746.2689747.

[29] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, und Liwu Chang. A novel anomaly detection scheme based on principal component classifier. In *Proceedings of International Conference on Data Mining*, 2003.

[30] Marina Sokolova, Nathalie Japkowicz, und Stan Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *AI 2006: Advances in Artificial Intelligence: 19th Australian Joint Conference on Artificial Intelligence, Hobart, Australia, December 4-8, 2006. Proceedings 19*, pages 1015–1021, 2006.

[31] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, und Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, 2020. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

[32] Tiago A. Almeida, Jose Maria Gomez Hidalgo, und Akebo Yamakami. Contributions to the study of sms spam filtering: New collection and results. In *Proceedings of the 2011 ACM Symposium on Document Engineering (DOCENG'11)*, 2011.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, und Illia Polosukhin. Attention is all you need, 2017.

[34] Abdeljalil Yaha. 3 months of nasdaq100 related news from gdelt. URL `https://www.kaggle.com/datasets/abdeljalilyahya/3-months-of-nasdaq100-related-news-from-gdelt`.

[35] Zengyou He, Xiaofei Xu, und Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9):1641–1650, 2003. ISSN 0167-8655. doi: 10.1016/S0167-8655(03)00003-5. URL `https://www.sciencedirect.com/science/article/pii/S0167865503000035`.

[36] Yue Zhao, Zain Nasrullah, und Zheng Li. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20(96):1–7, 2019. URL `http://jmlr.org/papers/v20/19-011.html`.

Abbildungsverzeichnis

Eidesstattliche Versicherung

(Affidavit)

Name, Vorname
(surname, first name)

Matrikelnummer
(student ID number)

☐ Bachelorarbeit
(Bachelor's thesis)

☐ Masterarbeit
(Master's thesis)

Titel
(Title)

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

A. Hüsler

Ort, Datum
(place, date)

Unterschrift
(signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:*

A. Hüsler

Ort, Datum
(place, date)

Unterschrift
(signature)

***Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.**