Angewandte Computer- und Biowissenschaften

HOCHSCHULE MITTWEIDA
University of Applied Sciences

MASTERTHESIS

Herr
Thomas Schäfer

Merkmale toxischer und aggressiver Kommentare in sozialen Netzwerken

und deren numerische Bewertung mithilfe von Deep Learning

2021

Fakultät **Angewandte Computer- und Biowissenschaften**

MASTERTHESIS

Merkmale toxischer und aggressiver Kommentare in sozialen Netzwerken

und deren numerische Bewertung mithilfe von Deep Learning

Autor:
Thomas Schäfer

Studiengang:
Cybercrime/Cybersecurity

Seminargruppe:
CY19wC-M

Matrikelnummer:
53137

Erstprüfer:
Prof. Dr. rer. nat. Dirk Labudde

Zweitprüfer:
Dr. rer. nat. Michael Spranger

Mittweida, Dezember 2021

Fakultät **Angewandte Computer- und Biowissenschaften**

MASTERTHESIS

Features of toxic and agressive social media comments

methods of numerical evaluation employing deep learning

Autor:
Thomas Schäfer

Studiengang:
Cybercrime/Cybersecurity

Seminargruppe:
CY19wC-M

Matrikelnummer:
53137

Erstprüfer:
Prof. Dr. rer. nat. Dirk Labudde

Zweitprüfer:
Dr. rer. nat. Michael Spranger

Mittweida, Dezember 2021

Bibliografische Angaben

Schäfer, Thomas: Merkmale toxischer und aggressiver Kommentare in sozialen Netzwerken, und deren numerische Bewertung mithilfe von Deep Learning, 115 Seiten, 19 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften

Masterthesis, 2021

Satz: LaTeX

Referat

In dieser Masterthesis wird evaluiert, wie gut sich Deep-Learning-Modelle für eine Toxizitätsbestimmung im digitalen Raum eignen. Hierfür wird die Transformer-Architektur anhand verschiedener Pre-Trainings auf BERT-, DistilBERT-, RoBERTa- und GPT-2-Basis mithilfe der toxisch-binär annotierten GermEval-Datensätze aus den Jahren 2018, 2019 und 2021 angepasst. Das Feintuning der Modelle findet sowohl mit Supervised-, als auch mit Semi-Supervised-Learning via GAN statt. Im Anhang dieser Arbeit steht der genutzte Programmcode zur Verfügung.

Das Feintuning via GAN stellt eine Besonderheit in der Herangehensweise automatisierter NLP-Aufgaben darf. Als Ergebnis dieser Arbeit kann deren Wirksamkeit in binären Textklassifizierungsaufgaben im deutschen Sprachraum bestätigt werden.

Onlinequellen wurden zum Zeitpunkt des Abrufs mithilfe des Firefox-Addons *"SingleFile"* in eine HTML-Datei gespeichert. Sowohl der HTML-Teil, als auch die Mediendateien, Stylesheets und Skriptdateien befinden sich komprimiert in der Datei. Jede Onlinequelle wurde während des Speichervorgangs bei woleet.io registriert, sodass später die Integrität der HTML-Datei geprüft werden kann. Hierfür speichert Woleet die Signatur und Zeitstempel einer Datei innerhalb der Bitcoin-Blockchain. Soll die Integrität einer Datei geprüft werden, kann dies über gildas-lormeau.github.io/singlefile-woleet/index.html erfolgen.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

ANN	artificial neural network
API	Application Programming Interface
AR	Augmented Reality
BERT	Bidirectional Encoder Representation from Transformers
bpb	Bundeszentrale für politische Bildung
CoBERL	Contrastive BERT for Reinforcement Learning
CPU	Central Processing Unit
DAU	Daily Active Users
dbmdz	Bayerische Staatsbibliothek - Münchener Digitalisierungszentrum
DL	Deep Learning
EOS	End of Sentence
FNN	Feedforward Neural Network
GAN	Generative Adversarial Network
GLUE	General Language Understanding Evaluation
GPT	Generative Pre-Training
GPU	Graphics Processing Unit
KI	Künstliche Intelligenz
KONVENS	Konferenz zur Verarbeitung natürlicher Sprache
LSTM	Long Short Term Memory
MAU	Monthly Active Users
MCC	Matthews correlation coefficient
ML	Machine Learning
MLFN	Multilayer Feedforward Neural Network
MLM	Masked Language Model
MSE	Mean Squared Error
NER	Named Entity Recognition
NetzDG	Netzwerkdurchsetzungsgesetz
NLP	Natural Language Processing
NSP	Next Sentence Prediction
OOM	out of memory

pip	Package installer for Python
PyPI	Python Package Index
QS	Qualitätssicherung
RACE	ReAding Comprehension from Examinations
RNN	Recurrent Neural Network
RoBERTa	Robustly optimized BERT pretraining approach
SemEval	Semantic Evaluation
SLFN	Single-Layer Feedforward Neural Network
SNN	simulated neural network
SOTA	state of the art
SQuAD	Stanford Question Answering Dataset
SVM	Support Vector Machine
TF-IDF	term frequency–inverse document frequency
TPU	Tensor Processing Unit
XL	Cross-Lingual

Teil I

Grundlagen und Recherche

1 Einleitung

> If your hate could be turned into electricity, it would light up the whole world.
>
> *Nikola Tesla (1856-1943)*

Wer einige Zeit in sozialen Medien verbringt, kann das Zitat von Nikola Tesla eventuell nachvollziehen, obwohl dieser nicht die sozialen Medien miterlebte. Während man in Konversationen von Person zu Person auch Hass erfahren kann, scheint in sozialen Netzwerken eine verachtende Atmosphäre keine Seltenheit zu sein. Dieses Phänomen ist unter anderem am Anstieg der Kriminalität im Bereich der Hassrede zu erkennen. [1] Um die sozialen Medien *“sozialer”* zu gestalten, gibt es verschiedene Ansätze. Einer von diesen ist die frühzeitige Entfernung rechtswidriger Inhalte. Dieser Ansatz soll in dieser Masterthesis behandelt werden. Deshalb ist das Ziel eine Erarbeitung und Evaluierung eines bestmöglichen Algorithmus zur binären Klassifizierung toxischer oder nichttoxischer Textinhalte. Verwandte Arbeiten stellen alle Einreichungen zu den in dieser Arbeit genannten GermEval-Herausforderungen dar.

1.1 Aufbau

Um die Zielstellung zu erfüllen, werden zunächst die Rahmenbedingungen dieser Arbeit erläutert. Die Themen in dieser Masterthesis werden hierbei immer spezifischer. Zuerst werden gesellschaftliche Fragestellungen, wie die Kritik, Terminologie und gesetzliche Rahmenbedingungen, behandelt. Weiter erfolgt eine Zusammenfassung zu den internationalen Forschungsbemühungen zum Thema Natural Language Processing (kurz: NLP).

Da der Schwerpunkt dieser Arbeit auf Deep Learning (kurz: DL) liegt, werden Architekturen dieser Künstlichen Intelligenzen (kurz: KI) erläutert. Auf dieser Grundlage wird ein Zwischenstand der aktuell bestgeeigneten DL-Modelle betrachtet. Mit diesen Informationen ist es Möglich einen Klassifikator für die definierte Aufgabe auszuwählen und an diese anzupassen. Abschließend werden die Ergebnisse eingeordnet und Empfehlungen für weitere Forschungen auf Basis dieser Arbeit gegeben.

1.2 Methoden der Sprachbewertung

Neben der manuellen Annotation durch Menschen werden diverse Klassifikatoren verwendet. Seit einigen Jahren spielt hierbei auch speziell trainierte KI eine große Rolle auf diesem Gebiet. Die von Menschen annotierten Daten können dazu dienen, KI oder andere Modelle zu trainieren. In dieser Masterthesis werden die einzelnen Modelle und die Funktionsweise der state of the art (kurz: SOTA) KI-Architekturen erläutert.

1.3 Kritik an automatisierter Sprachbewertung

Die Datenlast des Internets wird immer größer. Gleichzeitig verfügen immer mehr Menschen über einen Internetanschluss. [2] In Deutschland bestätigte der Bundesgerichtshof, dass ein funktionaler Internetanschluss als Grundrecht gilt. [3] Es ist kein Abwärtstrend der Datenlast des Internets absehbar. Darüber hinaus steigt die Anzahl der mit dem Internet zusammenhängenden Kriminalität von Jahr zu Jahr. [1, S. 10]

Ein Großteil der Internetnutzenden sieht Fake News und Hass als ernstzunehmende Bedrohung an. [4] Dem Diskurs im Internet wird ein generell rauerer Ton vorgeworfen. [5, S. 709] Twitter verzeichnet 750 Millionen Tweets [6] pro Tag, eine Datenmenge, deren manuelle Sichtung durch Menschen eine unmöglich erfüllbare Aufgabe ist. Aus diesem Grund wurden Modelle entwickelt, die bereits beim Absenden eines Kommentars prüfen, ob dieser gegen die Nutzungsbedingungen einer Plattform verstoßen. [7]

Ein Kritikpunkt dieser Algorithmen ist unter anderem, dass diese oftmals nicht transparent sind. Kritiker fordern deshalb eine Offenlegung der Quellcodes solcher automatisierten Verfahren der Sprachbewertung. Auch gibt es große Protestbewegungen gegen neue gesetzliche Bestimmungen der Vorfilterung von Inhalten. Ein Beispiel hierfür sind die Demonstrationen gegen den 13. Artikel der europäischen Urheberrechtsreform.

Die Bewegung gegen *“Artikel 13”* bemängelt unter anderem, dass Algorithmen auch Fehler begehen und zu Unrecht mehr Inhalte sperren, als es notwendig ist. Somit ist ein Plattformbetreiber zu Lasten der Nutzenden auf der sicheren Seite. Eine Befürchtung der Demonstrierenden ist eine Einschränkung der Meinungsfreiheit durch diese Algorithmen. Durch diese Praxis wird die Richtig-Positiv-Rate auf Kosten der Falsch-Positiv-Rate erhöht. Somit möchten sich Plattformbetreiber gegen empfindliche Bußgelder schützen. [8, S. 234] In der Sprachbewertung führt dies mitunter zur Falscherkennung sarkastischer Inhalte und Zitaten, dessen Problembewältigung essentiell für die weitere Forschung automatisierter Sprachbewertung ist. Die im Artikel 5 des Grundgesetzes definierte Meinungsfreiheit ist dennoch nicht mit einer Ausdrucksfreiheit zu verwechseln. So wird eine Aussage nicht durch die Meinungsfreiheit geschützt, wenn diese die persönliche Ehre von Menschen verletzt (Art. 5 Abs. 2 GG). [8, S. 234]

Wird ein Modell nicht moderiert trainiert, kann der Lernprozess sehr einseitig ausfallen. Im deutschen Sprachraum sind die Trainingsdaten beispielsweise noch nicht divers genug. Hierdurch entstehen besonders bei DL-Ansätzen nicht zufriedenstellende Ergebnisse, was die Fairness betrifft. [9] Um diesen Missstand zu beheben, ist es nötig sehr viel mehr annotierte Trainingsdaten für den deutschen Sprachraum bereitzustellen. [8, S. 244]

Unter anderem aus diesen Gründen zweifeln Forschende bei Facebook selbst an der Fähigkeit von KI Hass zu erkennen. So bringt laut Whistleblowerin Frances Haugen KI bei der Entfernung von Hassrede lediglich *"minimalen Erfolg"*. [10] Facebooks Vice President of Integrity, Guy Rosen, antwortete hierauf mit einem Blogpost zur Ausbreitung von Hassrede auf der Plattform. Diesem zufolge ist lediglich jeder 2.000. auf Facebook bereitgestellte Inhalt hasserfüllt. Demnach wurde innerhalb eines Jahres der gesehene Hassinhalt auf der Plattform halbiert. Rosen betont in diesem Blogpost mehrfach, dass die Entfernung von Hassinhalten keine zuverlässige Metrik sei, um den Erfolg von KI zu messen und die tatsächlich gesehenen Hassinhalte ausschlaggebender seien. [11] Die Zusammenarbeit zwischen Facebook und externen Prüfstellen ist nicht durchsichtig dokumentiert und lässt keine Prüfung der Messwertherkunft zu.

1.4 Probleme automatisierter Sprachbewertung

Die hohe Komplexität von Sprache, die Vielfalt der Sprachfamilien dieser Erde und der durch den Kontext bedingte Wandel der Sprachbedeutung erschweren den Prozess der automatisierten Sprachbewertung erheblich.

So kann in der deutschen Sprache ein Ausdruck über mehrere Bedeutungen verfügen. Hierbei spricht man von Äquivokation von Wörtern, die wiederum in Polysemie und Homonymie gegliedert ist. Polysemie steht für die Mehrdeutigkeit von Wörtern, bei denen die Bedeutungen ähnlich zueinander sind. Ein Beispiel für ein polysemietisches Wort ist *"Decke"*, welches als Stoff oder in einem Raum etwas von oben her begrenzt.

Homonymie beschreibt die Mehrdeutigkeit von Wörtern, bei denen die Bedeutungen keine Ähnlichkeiten aufweisen. Beispiele für homonyme Wörter sind unter anderem *"Bank"*, *"Schlange"* und *"Note"*. [12] Homonyme Wörter sind für die automatisierte Toxizitätserkennung dann problematisch, wenn nicht alle Bedeutungen den gleichen Toxizitätsgehalt aufweisen. Die Aussage, dass jemand wie ein nicht ästhetisches Tier aussehen soll, kann eine höhere Toxizität aufweisen als der Ausdruck des Tieres an sich. Dies macht im NOHATE-Datensatz mit 50% die absolute Mehrheit der False Negative Bewertungen aus. [9]

Bewerten Algorithmen einen Kommentar, können diese lediglich auf das trainierte Wissen zurückgreifen, um das Objekt zu bewerten. Die Erkennung von Sarkasmus ist oftmals kein Kriterium bei der Erstellung von Modellen. Diese Erkennung ist jedoch essentiell, da die

Einschätzung eines Kommentars sich drastisch verändern kann, sobald der Sarkasmus klassifiziert wurde. So kann ein sarkastischer Kommentar in der Sprache einem negativen Sentiment zugeordnet, in der Bedeutung jedoch einem positiven Sentiment zugeordnet werden und umgekehrt. [13, S. 2]

Aus diesen Gründen kann die Erkennung von Sarkasmus einen Genauigkeitsgewinn für die Toxizitätsbestimmung bedeuten. Wie stark dieser Genauigkeitsgewinn ausfällt hängt jedoch von der Toxizitätsdefinition ab, welche im folgenden Kapitel erläutert wird. Auch die Bedeutungsbestimmung homonymer Wörter kann über kontextuelle Vektoren drastisch verbessert werden. Kontextuelle Vektoren betrachten nicht nur einzelne Wörter, sondern verbinden mehrere Wörter zu einem Vektor. [14]

1.5 Terminologie

Hassrede

Was ist Hassrede (auch Hate Speech genannt) und was ist keine Hassrede? In Internetdiskussionen wird Hassrede unterschiedlich wahrgenommen. Jeder Nutzende verfügt über eine eigene Wahrnehmung von Hate Speech, wodurch eine eindeutige Definition für die Weiterbearbeitung dieser Masterthesis von Nöten ist. Da Nutzende sozialer Medien mit subjektiven Maßstäben urteilen, bedeutet Hate Speech für einige Nutzende bereits, dass gegen ihre eigene Meinung geschrieben wird. Die Nachrichten der Nutzenden, die diese Ansicht vertreten, fallen oftmals bei der Verbreitung von Hassrede auf. Für andere Nutzende ist Hassrede bereits jeder Kommentar, der weitere negative Kommentare auslöst. [5, S. 710]

Diese verschiedenen Interpretationsmöglichkeiten des Wortes *“Hassrede”* bilden einen Rahmen für dessen Definition. Die Bundeszentrale für politische Bildung (kurz: bpb) schreibt dem Begriff der Hassrede ebenfalls eine Unschärfe in der Definition zu. Eine Abgrenzung zwischen einer Beleidigung und Hassrede ist somit nicht immer klar durchführbar und oftmals auch für Wissenschaftler eine schwierige Aufgabe. [8, S. 234–235] Dies folgt aus der Ungenauigkeit der strafrechtlich relevanten Wörter.

“Wenn Menschen abgewertet, angegriffen oder wenn gegen sie zu Hass oder Gewalt aufgerufen wird, spricht man von Hate Speech.” [15] Somit ist die Hassrede vorwiegend ein Phänomen von gruppenbezogener Menschenfeindlichkeit, wie Rassismus, Antisemitismus, Sexismus. [8, S. 233] Diese trifft besonders im Internet, vorwiegend in sozialen Medien, auf. [16, S. 1] Oftmals liegt ein Tatbestand der Volksverhetzung vor. [15]

Sentiment

Das Wort Sentiment stammt aus der französischen Sprache und bedeutet *“Gefühl”*. Dieses wird oftmals in negativ, neutral und positiv gegliedert. [17, 18] Sentimentanalysen können mit aktuellen Technologien vielseitig eingesetzt werden. So werden nicht nur einzelne Social-Media-Auftritte auf deren Stimmungsverhältnis, sondern auch gesamte Gruppendynamiken untersucht. [19, 20] Hierbei wurde festgestellt, dass das Sentiment einer Gruppe sich auch auf die Arbeitsleistung auswirken kann. [21, S. 659]

Bis in die frühen 2000er-Jahre wurde eine Sentimentanalyse in den meisten Fällen manuell von Menschenhand betrieben. Um jedoch die immer größer werdenden Datenmengen bewältigen zu können, sind automatisierte Sentimentanalysen weitaus zeiteffizienter als manuelle Annotationen. [22]

Toxizität

Die Toxizität der Sprache steht für die Giftigkeit, über welche eine Aussage verfügt. Diese ist eine Zusammenstellung aus verschiedenen Faktoren.

Ein Faktor ist der Anteil an beleidigender Sprache. Beleidigende Sprache kann zielgerichtet verletzend, abwertend oder schmähend gegen eine Person gerichtet sein. [16, S. 1] Ein weiterer Faktor ist, wie sehr ein Kommentar dazu verleitet, weitere offensive Kommentare zu schreiben oder die Diskussion ohne Ergebnis zu verlassen. Toxizität ist nicht gleich Hassrede und nicht immer strafbar. Jedoch sind die Erkennungsmethoden ähnlich und lassen sich somit übertragen. [9]

Bei verletzender Sprache wird davon ausgegangen, dass bewusst eine Person beleidigt wird. Ein Indikator für abwertende Sprache ist, dass Beleidigungen und Schimpfwörter im Text enthalten sind, diese jedoch an konkrete Personen oder Gruppen gerichtet sind. Texte in dieser Kategorie können auch über ein positives Sentiment verfügen, da besonders in der Jugendsprache negative Wörter für die Darstellung positiver Sachverhalte verwendet werden. Im NOHATE-Datensatz sind etwa 60% der False Positives nicht-toxische Kommentare mit Schimpfwörtern. [9]

Verschmähende Kommentare richten sich gegen die Identität eines Menschen. Besonders Minderheiten sind betroffen von dieser Form beleidigender Sprache. Personen werden auf ihre Gruppenangehörigkeit reduziert und mit negativen Eigenschaften versehen, die diesen Gruppen zugeschrieben werden. Ob diese Eigenschaften wahr sind und ob diese Eigenschaften auf das beleidigte Individuum zutreffen, ist hierbei unerheblich. [23]

Das alles als Gesamtheit stellt im Kontext dieser Masterthesis den Begriff der Toxizität dar. Auf eine Detailauswertung der einzelnen Komponenten von Toxizität wird im Rahmen dieser Masterthesis bewusst verzichtet, da die Genauigkeiten der aktuellen SOTA-Lösungen in der deutschen Sprache gering sind. So liegen die sogenannten *"fine"*-Genauigkeiten bei er GermEval 2018 Challenge maximal bei 52,71% und bei der GermEval 2019 Challenge maximal bei 53,59%. [23]

Natural Language Processing

NLP beschreibt alle Vorgänge, bei denen computergestützt Text verarbeitet wird. Hierbei ist unerheblich, welche Technologie für diese Verarbeitung eingesetzt wird. [24] Die Aufgabe dieser Masterthesis stellt somit ebenfalls eine Aufgabe des NLPs dar.

Die bereits vorher beschriebene Sentimentanalyse ist ein großes Teilgebiet des NLPs. Ein weiteres Teilgebiet stellt die Named Entity Recognition (kurz: NER) dar. Bei dieser werden Objekte aus einem Text den zugehörigen Kategorien zugeordnet. So kann ein NER-Modell beispielsweise Namen, Orte, Tätigkeiten oder Uhrzeiten aus einem Text extrahieren.

Bei dem Topic Modeling findet ein Algorithmus Schlagwörter, um dem Text ein Thema zuzuordnen. Zur Zusammenfassung von langen Texten wird die Text Summarization verwendet. Mithilfe von Aspect Mining kann durch Kombination einer Sentimentanalyse und einem NER-Modell jedem Schlagwort ein Sentiment zugeordnet werden. So können wichtige Informationen von Rezensionen automatisiert zusammengefasst und ausgewertet werden. [24]

Künstliche Intelligenz

KI beschreibt alle Verfahren, bei welchen ein Computeralgorithmus verwendet wird, um ein Modell zu trainieren. Dieses Modell kann zur Umweltwahrnehmung und Entscheidungsfindung verwendet werden. Hierbei ist es notwendig, dass Eingangsdaten für das Modell vorhanden sind. Die KI muss sich also unter anderem auf die realitätsnahe Darstellung von Sensordaten oder anderen Eingangsgrößen verlassen. [25] In der nachfolgenden Grafik 1.1 wird die Beziehung zwischen KI, ML und DL visualisiert.

Machine Learning

Machine Learning (kurz: ML) verwendet vordefinierte Features, um ein Modell zu trainieren. Bei der Auswahl der auszuwertenden Features hat man durch die vielseitige Vorverarbeitungen im Erstellungsprozess einen signifikanten Einfluss. [25] Dies erleichtert eine Feature Extraction am fertigen Modell.

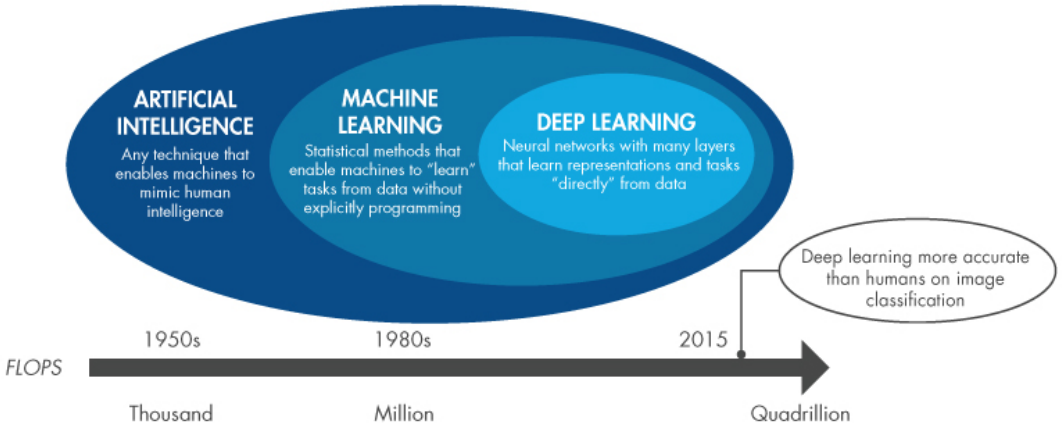

Abbildung 1.1: Beziehung zwischen KI, ML und DL [25]

Die Feature Extraction ist eine Methode im ML, mit welcher die aussagekräftigsten Eingangsgrößen ermittelt werden können. So kann unter anderem dargestellt werden, welche Wörter in einer Klassifizierungsaufgabe am wahrscheinlichsten zu einer bestimmten Zuordnung einer Klasse (Label) führen. Als weiteres Beispiels kann bei Bildklassifizierungsaufgaben ausgegeben werden, welche Pixel am wahrscheinlichsten welchem Objekt zugeordnet werden können.

Deep Learning

Wie in Abbildung 1.1 dargestellt, ist DL ein Teilgebiet des MLs. Das *"Deep"* steht hierbei für die versteckten Ebenen (auch Hidden-Layer), über welche ein solches Modell verfügt. [25] Die Ausführung dieser vieler Ebenen ist, je nach Modellgröße, deutlich rechenintensiver als die Anwendung eines ML-Modells.

Anders als bei dem herkömmlichen ML, arbeitet das DL näher am Datensatz. Es fallen vor dem Training einige Arbeitsschritte für den Menschen weg, sodass die Kontrolle über das Ergebnis bei DL geringer als bei ML ist. Während des Trainingsprozesses entscheidet das Modell mithilfe der Hidden-Layer, welche Features die aussagekräftigsten sind. [25]

DL erfordert eine enorme Menge an Trainingsdaten und Rechenleistung. Dementsprechend lohnt sich die Verwendung von DL nur, wenn Zugriff zu sehr vielen (annotierten) Trainingsdaten besteht. Aus diesem Grund ist eine DL-Lösung häufig kostenintensiver als eine reine ML-Lösung. Durch die komplexe Architektur des fertigen Modells kann die Feature Extraction bei einem DL-Modell mit einem höheren menschlichen Zeitaufwand als bei einem ML-Modell ermittelt werden.

Begriffe und Methoden der Statistik

In der Modellevaluierung werden einheitliche Genauigkeitsmetriken benötigt. Daher sind in Tabelle 1.1 verschiedene Berechnungsmethoden der Modellgenauigkeit dargestellt. Bevorzugt werden der Macro-F1-Score und der Matthews correlation coefficient (kurz: MCC) für eine möglichst genaue Darstellung der Modellgenauigkeit verwendet. Diese betrachten die True und False Negatives und Positives. Zusätzlich normalisieren Macro-F1 und MCC die Klassengrößen für eine bessere Messung der Modellgenauigkeit. [26]

Wert	Symbol oder Formel	Definition
True Positive	tp	Richtig positiv klassifizierte Daten
True Negative	tn	Richtig negativ klassifizierte Daten
False Positive	fp	Negative Daten positiv klassifiziert
False Negative	fn	Positive Daten negativ klassifiziert
Accuracy [27]	$\frac{tp+tn}{tp+tn+fp+fn}$	Anteil richtig klassifizierter Daten
Precision [27]	$\frac{tp}{tp+fp}$	Anteil richtig klassifizierter Positive
Recall [27]	$\frac{tp}{tp+fn}$	Sensitivität des Modells
F1 [27]	$2 * \frac{precision*recall}{precision+recall}$	Harmonisches Mittel zwischen Precision und Recall
Macro-F1 [27]	$\frac{1}{N}\sum_{i=0}^{N} \mathrm{F1}_i$	F1 - Gleiche Wichtung aller Klassen
MCC [26]	$\frac{tp*tn-fp*fn}{\sqrt{(tp+fp)(tp+fn)(tn+fp)(tn+fn)}}$	Normalisierung der unterschiedlichen Klassengrößen

Tabelle 1.1: Genauigkeitsmetriken für ML-Modelle

1.6 Gesetzesgrundlagen

Laut einer Auswertung einer Langzeitstudie der Johannes Gutenberg Universität Mainz aus dem Jahr 2018 befürwortet ein Großteil der Internetnutzenden eine stärkere Ahndung von Hassrede im Internet. [4] Hierfür gibt es bereits einige Gesetzesgrundlagen. In der folgenden Liste sind strafbare Sachverhalte zusammengestellt, die im Zusammenhang zu Hassrede stehen:

- § 86 StGB Verbreiten von Propagandamitteln verfassungswidriger Organisationen [28]
- § 86a StGB Verwenden von Kennzeichen verfassungswidriger Organisationen [28]
- § 90a StGB Verunglimpfung des Staates und seiner Symbole [28]
- § 90b StGB Verfassungsfeindliche Verunglimpfung von Verfassungsorganen [28]

- § 91 StGB Anleitung zur Begehung einer schweren staatsgefährdenden Gewalttat [5, S. 707]
- § 100a StGB Landesverräterische Fälschung [5, S. 707]
- § 111 StGB Öffentliche Aufforderung zu Straftaten [15]
- § 126 StGB Störung des öffentlichen Friedens durch Androhung von Straftaten [28]
- § 129 – § 129b StGB Bildung krimineller Vereinigungen, Bildung terroristischer Vereinigungen, Kriminelle und terroristische Vereinigungen im Ausland [5, S. 707]
- § 130 StGB Volksverhetzung [15]
- § 131 StGB Gewaltdarstellung [28]
- § 140 StGB Belohnung und Billigung von Straftaten [28]
- § 166 StGB Beschimpfung von Bekenntnissen, Religionsgesellschaften und Weltanschauungsvereinigungen [28]
- § 184b StGB Verbreitung, Erwerb und Besitz kinderpornographischer Schriften [5, S. 708]
- § 185 StGB Beleidigung [15]
- § 186 StGB Üble Nachrede [28]
- § 187 StGB Verleumdung [15]
- § 189 StGB Verunglimpfen des Andenkens Verstorbener [28]
- § 201a StGB Verletzung des höchstpersönlichen Lebensbereichs und von Persönlichkeitsrechten durch Bildaufnahmen [28]
- § 240 StGB Nötigung [15]
- § 241 StGB Bedrohung [15]
- NetzDG [8, S. 233]
- Art. 5 GG Recht auf freie Meinungsäußerung [8, S. 234]

Trotz dieser Gesetzesgrundlagen fällt Hassrede nicht immer in den strafvollzugsfähigen Bereich. Gerade im Bezug auf das im Jahr 2017 erlassene Netzwerkdurchsetzungsgesetz (kurz: NetzDG) liegt die Verantwortung in der Entfernung von Hassrede bei den Plattformbetreibenden. Die hierbei einzuhaltende Frist beträgt 24 Stunden ab der Veröffentlichung des rechtswidrigen Inhalts. [8, S. 234]

2 Anatomie eines Social-Media-Beitrags

Die Vielfalt der Social Media Beiträge steigt stetig. Wo Facebook früher mit einem simplen Angebot den Markt dominierte, herrscht nun ein reger Wettkampf um die Aufmerksamkeit der Internetnutzenden. Deshalb verfügen die großen Plattformen mittlerweile über mehr und komplexere Funktionen als vor noch einem Jahrzehnt.

Ferner werden in dieser Masterthesis nur Social-Media-Plattformen behandelt, die mindestens über eine direkt mit dem Desktopcomputer auslesbare Kommentarfunktion verfügen. Dies soll den Umfang dieser Masterthesis auf einen realistischen Rahmen einschränken.

2.1 Historie

Facebook gilt seit dem Gründungsjahr 2004 als das größte Soziale Netzwerk der Welt. [29] Facebook verzeichnet zum Zeitpunkt des Verfassens dieser Arbeit knapp zwei Milliarden Daily Active Users (kurz: DAU) und knapp drei Milliarden Monthly Active Users (kurz: MAU). [30]

Die Videoplattform YouTube baute das Programm bereits im Gründungsjahr 2005 deutlich aus und ist seither eine der am schnellsten wachsenden Internetseiten. Bereits ein Jahr nach der Gründung YouTubes beträgt die Anzahl der täglich wiedergegebenen Videos auf der Plattform bereits etwa 100 Millionen Videos. Ebenfalls im Jahr 2006 übernahm Google YouTube. Die Videoplattform wird seither als Tochterunternehmen von der Google LLC gehalten. [31]

Instagram ist eine vorrangig auf mobilen Endgeräten genutzte Plattform, deren Gründung das Jahr 2010 datiert. Bereits im Jahr 2012 übernahm Facebook für 737 Millionen US-Dollar die Bildplattform. Zu diesem Zeitpunkt verzeichnete Instagram etwa 30 Millionen registrierte Nutzende, während Facebook bereits im gleichen Zeitraum 955 Millionen registrierte Nutzende verzeichnete. [32] Durch Instagram wurden Beiträge, die fast ausschließlich ihre Inhalte über ein Bild kommunizieren, immer populärer. Instagram ist bis in der Mitte des Jahres 2018 bereits auf eine Größe von einer Milliarde Konten herangewachsen. [33]

Snapchat bietet Bild- und Videoinhalte an, welche nach 24 Stunden oder nach einer Betrachtung wieder in der App verschwinden und auf dem gesamten Handybildschirm dargestellt werden. Dies hat zur Folge, dass der Handybildschirm effektiver mit Beitragsinhalt gefüllt wird. Die gesendeten und empfangenen Elemente verschwinden jedoch nicht von den Endgeräten und lassen sich noch aus dem Dateisystem auslesen. [34]

Außerdem implementierte Snapchat als erste große Plattform ein konsequentes Angebot von Augmented-Reality (kurz: AR). Snapchat verzeichnet zum Zeitpunkt des Verfassens dieser Masterthesis laut Finanzreport 293 Millionen DAU. [35]

TikTok, ursprünglich eine Plattform für durch die einfache Erstellung von Lippensynchronisationsvideos, ist heute eine der beliebtesten Apps weltweit. TikTok steht häufig in der Kritik, da Sicherheitslücken nicht zeitnah geschlossen werden und Menschenrechte auf der Plattform nicht umgesetzt werden. So ist unter anderem der Einsatz für Rechte Nichtheterosexueller Menschen auf der Plattform verboten. [36, 37] Im Juli 2020 verzeichnete TikTok weltweit 689 Millionen MAU. [38]

Um ein Abwandern zu anderen Plattformen zu verhindern, integrieren Plattformbetreibende in ihre Anwendungen die Funktionen der Konkurrenz ein. Um Instagram konkurrenzfähig mit YouTube machen zu können, wurde *“IGTV”* geschaffen. So konnte statt der bisher üblichen 60 Sekunden möglichen Videolänge bis zu 60 Minuten Videomaterial hochgeladen werden. Auf IGTV sollen die Videos, anders als bei YouTube, jedoch im Hochformat bereitgestellt werden. [33]

Die von Snapchat bekannten Formate finden sich mittlerweile als sogenannte *“Stories”*, *“Statusmeldungen”* oder *“Series”* unter anderem auch auf Facebook, WhatsApp, Instagram, YouTube, Medium, Google AMP und auch dem Karrierenetzwerk LinkedIn wieder. Diese Ersatzprodukte zur Snapchat-Story sind oftmals in einem sehr geringen Umfang implementiert und reichen oft nicht über die Implementierung von Text und Bildern hinaus. Eine fertig erstellte Story verfügt, außer bei Google AMP, nicht über herauskopierbare Texte. Auch Snapchats AR-Filter wurden von Instagram übernommen. Die auf TikTok populären Kurzvideos finden sich heute bei Instagram als *“Reels”* und bei YouTube als *“Shorts”* wieder.

Die hierdurch entstandene Komplexität der Plattformen erfordert eine komplexere Analyse der Inhalte, als vor zehn Jahren der Fall war. Die Änderungen an den Plattformen treten so häufig auf, dass in immer kürzeren Zeiträumen evaluiert werden muss, ob die eingesetzten Auswertungsmethoden noch repräsentativ für die einzelnen Plattformen sind.

2.2 Merkmale

Jeder Beitrag in einem sozialen Netzwerk verfügt über Merkmale. Diese sind alle in dem Beitrag vorhandenen Daten, inklusive aller Metadaten. Je nach Plattform unterscheiden sich die für die Nutzenden möglichen und ersichtlichen Daten.

Wie bereits im vorherigen Kapitel erläutert, können Plattformen mitunter mehrere Möglichkeiten des Verbreitens von Informationen anbieten. Bei Facebook und Instagram wird

zwischen einem regulären Beitrag (Post), einer Story und einem Kommentar (Comment) unterschieden. Reddit bietet keine Stories an. Bei Twitter gibt es Tweets und Antworten auf Tweets (Replies). Diese Replies unterscheiden sich in der möglichen Darstellung nicht von einem Tweet.

In der Tabelle 2.1 sind die Möglichkeiten der einzelnen Beitragsformen der größten Social-Media-Plattformen mit einem großem auswertbaren Textanteil enthalten.

	Twitter		Facebook			Instagram			Reddit	
Merkmal	Tweet	Reply	Post	Story	Comment	Post	Story	Comment	Post	Comment
Name	X	X	X	X	X	-	-	-	-	-
Handle	X	X	-	-	-	X	X	X	X	X
Profilbild	X	X	X	X	X	X	X	X	-	X
Ort	-	-	X	-	-	X	X	-	-	-
Mention	X	X	X	-	X	X	X	X	X	X
Hashtag	X	X	X	-	X	X	X	X	-	-
Zitat (teilen)	X	X	X	-	-	-	X	-	X	X
Umfrage	X	X	-	-	-	-	X	-	X	-
Live-Video	-	-	X	-	-	-	X	-	X	-
Spenden	-	-	X	-	-	-	-	-	-	-
Q&A	-	-	X	-	-	-	X	-	-	-
Zeitangabe	genau	genau	genau	ungenau	genau	ungenau	ungenau	ungenau	genau	genau
Textzeichen	280	280	63.206	63.206	8.000	2.200	-	2.200	40.000	10.000
Emojis	X	X	X	X	X	X	X	X	X	X
Bild	X	X	X	X	X	X	X	-	X	-
Galerie	X	X	X	-	-	X	-	-	X	-
GIF	X	X	X	X	X	-	X	-	X	-
Sticker	-	-	-	-	X	-	X	-	-	-
Video	X	X	X	X	X	X	X	-	X	-
Audio	-	-	-	-	-	-	-	-	-	-
Gefühl	-	-	X	-	-	-	-	-	-	-
Like	X	X	X	-	X	X	-	X	X	X
Dislike	-	-	-	-	-	-	-	-	X	X
Emotion	-	-	X	X	X	-	X	-	-	-

Tabelle 2.1: Anatomie von Inhalten in sozialen Netzwerken

2.3 Kontextmerkmale

Alle Informationen, die nicht aus den Daten eines einzelnen Beitrags ersichtlich sind, zäheln zu den Kontextmerkmalen. Diese sind direkt mit dem Beitrag zusammenhängende Daten, wie beispielsweise die vorherigen Kommentare und der Beitrag an sich. Die Profilbilder und Namen der Kommentierenden zählen ebenfalls zu den Kontextmerkmalen. Des Weiteren können auch das Profil der ursprünglich verfassenden Person und aller Kommentatoren als Kontextmerkmal eines Kommentars verwendet werden. Ferner zählen die politische Situation und Trends zum Zeitpunkt des Verfassens der Texte eine Rolle für die Bewertung dieser.

Durch die Einbeziehung von Umweltfaktoren können ansonsten nicht-toxische Wortkombinationen plötzlich eine toxische Wirkung aufweisen. So kann eine Gutheißung von Aktionen einer bekannten Persönlichkeit toxisch sein, wenn diese für Gesellschaftsfeindlichkeit steht. Eine automatisierte Auswertung dieser Kontextmerkmale ist bisher

lediglich bedingt möglich. Analysen über den gesamten Kommentarverlauf können eine sinnvolle Ergänzung der Merkmale eines Beitrags darstellen. Darüber hinaus können Namen und Profilbild und, je nach Plattform, auch die Namen der Kommentierenden für die Verarbeitung verwendet werden. Das Profil einer Person kann dazu dienen, die durchschnittliche Toxizität aller Beiträge zu ermitteln. Eine automatische Auswertung von Texten unter Berücksichtigung der politischen Situation und von Trends kann jedoch nur mit einem enormen Aufwand betrieben werden. Eine umfassende automatisierte Analyse mit diesen Umweltfaktoren in Bezug auf Toxizität konnte in der Recherche für diese Masterthesis nicht ausfindig gemacht werden.

3 Untersuchungen zur Toxizitätserkennung mithilfe von Machine Learning

Die automatisierte Erkennung von toxischen Inhalten ist keine neue Aufgabenstellung. Trotz vieler Forschungsbemühungen erreichen automatisierte Modelle bei NLP-Aufgaben noch nicht die Präzision geschulter Menschen. Die meisten Forschungsergebnisse lernen die Modelle mit englischsprachigen Trainingsdaten an. Die dabei entstandenen SOTA-Lösungen werden von verschiedenen Forschungsgruppen in die jeweiligen Muttersprachen übertragen.

3.1 GermEval

Zu der größten deutschsprachigen Toxizitätsforschung zählt GermEval. GermEval ist ein Teil der Konferenz zur Verarbeitung natürlicher Sprache (kurz: KONVENS), welche jährlich in der DACH-Region stattfindet. Die KONVENS wird von der Sektion Computerlinguistik der Deutschen Gesellschaft für Sprachwissenschaft, der Gesellschaft für Sprachtechnologie und Computerlinguistik und der Österreichischen Gesellschaft für Artificial Intelligence organisiert. [39]

Die Herausforderungen der einzelnen GermEval-Ausgaben werden zu jeder KONVENS neu ausgeschrieben. Diese Herausforderungen sind vielseitig angelegt. Alle verfolgen jedoch das übergeordnete Ziel, die deutsche Sprache bewertbar zu machen. Zu den zentralen Forschungsfeldern zählt deshalb neben der Informatik auch die Linguistik. [16] Im Folgenden werden die GermEval-Aufgaben betrachtet, bei welchen binäre Label zwischen *“toxisch”* und *“nicht-toxisch”* zu klassifizieren sind. Zu den Ergebnissen der einzelnen GermEval-Ausgaben finden in der Modellevaluierung Vergleiche mit dem in dieser Arbeit trainierten Modell statt.

GermEval 2018

Die *“GermEval 2018 Shared Task on the Identification of Offensive Language”* ist die erste von der Konferenz ausgeschriebene Aufgabe zur Erkennung toxischer Sprache. Darüber hinaus ist es die erste ausgeschriebene Aufgabe im deutschen Sprachraum mit einer vergleichbaren Menge an annotierten Trainings- und Testdaten. [23, S. 352] Bei der Aufgabe handelt es sich um die Erkennung toxischer Sprache in zwei verschiedenen Teilaufgaben. Die erste Teilaufgabe besteht in der binären Erkennung toxischer Sprache. Die zweite Teilaufgabe fordert eine feinere Gliederung in die Unterkategorien toxischer Sprache. Hierfür wurden die Kategorien *“profanity”*, *“abuse”* und *“insult”* vorgegeben. Für diese Aufgabenstellungen haben sich insgesamt 20 Teams eingeschrieben. [16, S. 1]

Der Korpus für beide Teilaufgaben unterscheidet sich lediglich in den verfügbaren Spalten der Annotationen. [16, S. 3] Für die Beschaffung der Trainingsdaten wurden 8541 Tweets gesammelt und von drei Mitorganisierenden annotiert. Damit die Annotationsergebnisse unter den Mitorganisierenden auf Übereinstimmung geprüft werden können, wurden 300 Tweets von allen drei Personen bewertet und gegengeprüft. Für das Annotationsteam gänzlich unverständliche Tweets wurden aus dem Datensatz entfernt. [16, S. 4]

Der hierbei entstandene Korpus beinhaltet 5.009 Trainingsdaten. Von diesen sind 1.688 Hasskommentare und 3.321 neutrale Kommentare. Unter den Hasskommentaren befinden sich 1.022 in der Kategorie *“abuse”*, 595 in der Kategorie *“insult”* und 71 in der Kategorie *“profanity”*. Die Labels sind eindeutig, weshalb kein Kommentar über mehrere Labels verfügen kann. [16, S. 5]

Der Testdatensatz besteht aus 3.532 Tweets. Von diesen sind 1.202 Hasskommentare und 2.330 neutrale Kommentare. Unter den Hasskommentaren befinden sich 773 in der Kategorie *“abuse”*, 381 in der Kategorie *“insult”* und 48 in der Kategorie *“profanity”*. [16, S. 5]

Für die Evaluierung eines im Rahmen dieser Masterthesis angepassten Modells werden die annotierten Daten der ersten Teilaufgabe verwendet.

GermEval 2019

Die *“GermEval 2019 Shared Task 2”* dient zur weiterführung der Aufgabe des Jahres 2018. Die Aufgabenstellung ist die gleiche, jedoch erweitert die Ausgabe des Jahres 2019 den gesamten Datensatz auf ungefähr 12.000 Tweets. [23, S. 352–353]

Der hierbei entstandene Korpus beinhaltet 3.994 Trainingsdaten. Von diesen sind 1.287 toxische und 2.707 neutrale Kommentare. Unter den toxischen Kommentaren befinden sich 510 in der Kategorie *“abuse”*, 625 in der Kategorie *“insult”* und 152 in der Kategorie *“profanity”*. Der Testdatensatz besteht aus 3.031 Tweets. Von diesen sind 970 toxische und 2.061 neutrale Kommentare. Unter den toxischen Kommentaren befinden sich 400 in der Kategorie *“abuse”*, 459 in der Kategorie *“insult”* und 111 in der Kategorie *“profanity”*. [23, S. 357]

Im Rahmen dieser Masterthesis wird die Teilaufgabe 1 verwendet, bei welcher eine Grobannotation binär zwischen toxischer und nicht-toxischer Sprache unterscheidet.

GermEval 2021

Die *“GermEval 2021 Shared Subtask 1 - Toxic Comment Classification (Binary Classification Task)”* beinhaltet Daten, die von anonymisierten Facebook-Kommentaren einer

deutschen Fernsehtalkshow gesammelt wurden. Alle Kommentare stammen aus einem Zeitraum zwischen Februar und Juli 2019. Ferner beinhaltet die GermEval-Challenge 2021 Annotationen für Kommentare mit einem Sentiment positiver Diskussionsgrundlage. Hierbei handelt es sich um *"Subtask 2: Engaging Comment Classification"*. Bei der *"Subtask 3: Fact-Claiming Comment Classification"* handelt es sich um annotierte Kommentare, die eine Gegebenheit als korrekten Fakt angeben. Dies soll in als Vorverarbeitungsschritt zur Erkennung von Falschmeldung dabei helfen, potentiell zu analysierende Kommentare vorzufiltern. [40]

Der hierbei entstandene Korpus beinhaltet 3.244 Trainingsdaten. Von diesen sind 1122 toxische und 2122 neutrale Kommentare. Der Testdatensatz besteht aus 944 Kommentaren. Von diesen sind 504 toxische und 440 neutrale Kommentare. [40, S. 4] Die Subtask 1 von GermEval 2021 wird für Modellevaluierungen in dieser Arbeit verwendet.

3.2 SemEval

Bei SemEval (kurz: Semantic Evaluation) handelt es sich um eine Konferenz für Methoden der Sprachanalyse im englischen Sprachraum. Die bisherigen Konferenzen fanden in nordamerikanischen, europäischen und asiatischen Großstädten statt. Hierbei wird SemEval immer als Co-Veranstaltung einer anderen großen NLP-Veranstaltung ausgetragen. Bereits am 6. August 2021 wurde die 16. Iteration von SemEval (2022) angekündigt. Einen Monat darauf wurden die Trainingsdaten für alle zwölf Aufgaben veröffentlicht. [41]

SemEval 2021 Aufgabe 5 stellt eine binäre Klassifizierungsaufgabe zwischen toxischen und nicht toxischen Textausschnitten dar. Die verwendeten Methoden der fünf Teams mit der höchsten erreichten Genauigkeit sind RoBERTa, ERNIE und BERT mit Genauigkeiten zwischen 69% und 71%. Die besten Ergebnisse wurden demnach mit DL-Modellen erzielt. Vereinzelte Teams nutzten Support Vector Machines, (kurz: SVM) um das DL-Modell zu unterstützen. Unter den besten fünf Teams war jedoch keine SVM dabei. [42]

Dies ist ein Indikator für die hohe Komplexität einer binären Klassifizierungsaufgabe von Toxizität, welche bereits durch die GermEval-Konferenz aufgezeigt wurde. Durch diese hohe Komplexität ist die Datenmenge erreicht, ab welcher sich DL-Ansätze gegenüber klassischem ML durchsetzen können.

3.3 Google Jigsaw

3.3.1 Motivation

Google ist auf dem Gebiet des DLs ein technologischer Vorreiter. Das Unternehmen gründete hierfür die Abteilung *"Google AI"*. Die ML-Entwicklungsplattform *"TensorFlow"* ist ein Teil von Google AI. Forschungsteams von Google sind erfolgreich in vielen verschiedenen Kategorien der automatisierten Datenverarbeitung tätig. Forschungsgebiete umfassen unter anderem die Bild- und Videoverarbeitung, Sprachübersetzung & -analyse, Data Mining, E-Commerce, Kunst und viele weitere. [43]

Neben Google AI wurde Google Jigsaw gegründet. Jigsaw ist eine Arbeitsgruppe mit dem Ziel er sicheren Gestaltung des Internets. Die Aufgaben von Jigsaw sind die Bekämpfung von Desinformation, Zensur, Toxizität und gewalttätigem Extremismus. [44]

3.3.2 Ergebnis

Google Jigsaw schuf die Perspective API. Für die Entwicklung sammelte Google Jigsaw in öffentlichen Challenges eine Vielzahl von Datensätzen. Auf Kaggle ist ein Teil der für die Perspective-Entwicklung verwendeten Trainingsdatensätze von insgesamt 7 GiB abrufbar. Unter diesen Daten befinden sich ausschließlich englischsprachige Trainingsdaten. Auch die multilinguale Jigsaw-Herausforderung auf Kaggle bietet lediglich englischsprachige Trainingsdaten. Dies begründet Google Jigsaw damit, dass nicht die Trainingsdaten multilingual sind, sondern das Modell in seiner definierten Form allgemeingültig auf alle Sprachen anwendbar sein soll. [45]

Wie das Modell hinter Perspective funktioniert, gibt Google Jigsaw selbst nicht an. Auf der Webseite der Perspective API ist lediglich zu finden, dass es sich um *"Machine Learning"* handelt. [46] Auch die GitHub-Seite der Perspective API dient lediglich zur Dokumentation von Anwendungszwecken. Informationen zur Funktionsweise des Modells hinter der API werden nicht vermittelt. [47]

Die Veröffentlichung der Perspective API geschah Anfang 2017. [48] Ab diesem Zeitpunkt ist es mit einem kostenlosen API-Key möglich eine Anfrage pro Sekunde von Perspective auswerten zu lassen. Sollten mehr als eine Anfrage pro Sekunde nötig sein, lässt sich die Zeitbegrenzung auf persönliche Nachfrage bei Google erhöhen. In der FAQ weist Google Jigsaw darauf hin, dass Perspective durch die aktuelle Präzision nicht in der Lage ist, ohne Aufsicht eines moderierenden Menschens, Entscheidungen zu treffen. Eine Toxizität wird von der Perspective API mit einer Zahl im Intervall $[0, 1]$ beschrieben. [49]

3.3.3 Anwendung

Bevor die erste Anfrage an Perspective gesendet werden kann, muss ein API-Key erworben werden. In der Google Cloud Platform muss dieser Schlüssel für ein Projekt per Formular beantragt werden. Der API-Key wurde im Fall dieser Masterthesis innerhalb von einer Stunde von Google bereitgestellt.

Die Perspective API kann über HTTP-POST-Anfragen angesprochen werden. Die Dokumentation der Perspective API stellt Codebeispiele für die Betriebssystems-, Python- und Node.js-Umgebung bereit. Somit besteht die Möglichkeit die Daten direkt über curl auszuwerten. Für eine Analyse auf Webserverebene bietet Google das googleapis-Paket für Node.js an. Werden Anfragen in Python benötigt, muss vorher ein Teil des googleapiclient mit dem Namen discovery importiert werden.

Eine Abfrage der Perspective API mit einem deutschen Kommentar, bei welchem Google die Anfrage nicht speichert (`doNotStore`), benötigt in Python folgenden Code:

```
#!/bin/python
from googleapiclient import discovery
import json

API_KEY = 'API_KEY'
client = discovery.build(
  "commentanalyzer",
  "v1alpha1",
  developerKey=API_KEY,
  discoveryServiceUrl="https://commentanalyzer.googleapis.com/
   $discovery/rest?version=v1alpha1",
  static_discovery=False,
)
analyze_request = {
  'comment': { 'text': 'Hallo Welt!' },
  'requestedAttributes': {'TOXICITY': {}},
  'languages': 'de',
  'doNotStore': True
}
response = client.comments().analyze(body=analyze_request).execute
   ()
print(json.dumps(response["attributeScores"]["TOXICITY"]["
   summaryScore"]["value"]))
```

```
>> 0.03461841
```

Die Toxizität des Textes *“Hallo Welt!”* beträgt laut der Perspective API dementsprechend etwa 3,5%.

3.4 Challenges auf GitHub

GitHub ist laut alexa.com der beliebteste Ort für Open-Source-Entwicklung und auf Platz 40 der meist besuchten Internetseiten weltweit (Stand: Mitte 2019). [50] Im Folgenden werden die drei Projekte zur Toxizitäts- oder Hassredenerkennung vorgestellt, die auf der Plattform am beliebtesten sind. Für die Beliebtheit der Projekte werden die Sternmarkierungen und die Forks als Maßstab verwendet.

unitaryai/detoxify

Detoxify erschien im September 2020 auf GitHub und wird bis zum Zeitpunkt des Verfassens dieser Masterthesis aktiv weiter entwickelt. Das Ziel von Detoxify ist es, ein Rahmenwerk für die meistverwendeten englischsprachigen DL-Modelle zu schaffen, um mit diesen an den drei Jigsaw-Herausforderungen teilzunehmen. Detoxify unterstützt die `bert-base-uncased`, `roberta-base` und `xlm-roberta-base` Modelle.

Detoxify befindet sich im *"PyPI"*-Repository und kann somit über `pip` installiert werden. Sind alle Abhängigkeiten installiert, kann im Stammverzeichnis des GitHub-Projekts `python run_prediction.py --input 'Hello World!' --model_name original` ausgeführt werden. Beim ersten Ausführen lädt das Python-Programm das Modell von torch.hub herunter, bevor es mit der Evaluierung beginnt. Ausgabewerte des Modells sind *"toxicity"*, *"severe_toxicity"*, *"obscene"*, *"threat"*, *"insult"* und *"identity_hate"*. [51]

Soll detoxify in ein Python-Programm integriert werden, kann es mit dem PyPI-Paket importiert werden. Die Eingangsdaten können als String oder eine Liste von Strings an detoxify übergeben werden. Eine Ausgabe mit pandas verhilft bei einer Inputliste zu einer deutlich besseren Übersichtlichkeit.

```
#!/bin/python
from detoxify import Detoxify
import pandas as pd
input_text = ['Hello World!','Hallo Welt!']
results = Detoxify('original').predict(input_text)
print(pd.DataFrame(results,index=input_text).round(4))
```

```
>>              toxicity  severe_tox.  obscene  threat  insult  identity_hate
Hello World!  0.0011       0.0001     0.0002   0.0001  0.0002         0.0001
Hallo Welt!   0.0361       0.0001     0.0009   0.0003  0.0010         0.0005
```

Um auf den vereinfachten Trainingsprozess von Modellen über detoxify zugreifen zu können, müssen sich die drei auf Kaggle verfügbaren Herausforderungen von Jigsaw in einem neu angelegten *jigsaw-data*-Ordner befinden. Nachdem mit *create_val_set.py* ein Validierungsdatensatz aus den Trainingsdaten extrahiert wurde, kann mit der *train.py*-Datei und einer Trainings-*.json*-Datei mit den benötigten Hyperparametern ein Modell trainiert werden. [51]

Das Team hinter detoxify nutzt die durch das Projekt gewonnene Reichweite, um auf die Risiken von DL aufmerksam zu machen. So herrscht das Problem, dass DL-Algorithmen nur so gut sind, wie ihre Trainingsdaten. Wird dem Lernalgorithmus ein Datensatz präsentiert, der vorwiegend einer Ethnie oder einer politischen Gruppe entstammt, werden Vorhersagen des fertigen Modells ebenfalls eine Tendenz zu den unausgeglichen trainierten Datenmengen zeigen. [52]

Ein Beispiel hierfür ist der Genauigkeitsunterschied von Gesichtserkennungssoftware, abhängig von Hautfarbe und Geschlecht. Deshalb können KI-Modelle Menschen durch die Verarbeitungsergebnisse diskriminieren, wenn es sich beispielsweise um Strafverfolgung oder Bewerbungsverfahren handelt. [53]

zake7749/DeepToxic

DeepToxic ist, wie detoxify ebenfalls, eine auf GitHub veröffentlichte Einreichung einer auf Kaggle ausgetragenen Jigsaw-Herausforderung. Anders als detoxify greift DeepToxic nicht auf BERT, RoBERTa oder XLNet zu. Stattdessen wird bei DeepToxic ein komplett eigenes Modell trainiert, was ausschließlich dem Zweck der Toxizitätsbestimmung in der englischen Sprache dient. [54]

Das Modell hinter DeepToxic ist ein Recurrent Neural Network (kurz: RNN). Ein RNN ist in der Lage Informationsketten zu verarbeiten. Im Fall der NLP ist dies entweder eine Folge von Buchstaben oder Wörtern. Die genaue Funktionsweise von DL-Modellen auf RNN-Basis wird in dem Kapitel 4.2.3 erläutert.

In dem git-Repository sind alle Funktionen für das Training und Anwenden des Modells. Jedoch sind die Funktionen nicht dokumentiert [54], was das Nachvollziehen dieser verkompliziert. Eine Migration der Erkenntnisse dieser undokumentierten Funktionen in den Kontext dieser Masterthesis wird aus diesem Grund nicht durchgeführt.

pinkeshbadjatiya/twitter-hatespeech

Twitter-hatespeech ist ein Projekt zweier Sprachwissenschaftler, welches zuletzt im Dezember 2017 aktualisiert wurde. [55] Somit ist twitter-hatespeech ein Jahr vor BERT erschienen. Zu beobachten ist, dass vor dem Erscheinen von BERT viele verschiedene Architekturen verwendet wurden, welche seit dem Erscheinen von BERT deutlich seltener in wissenschaftlichen Publikationen erwähnt werden.

Genau wie bei DeepToxic kann bei twitter-hatespeech ein RNN-basiertes Modell verwendet werden. Dieses basiert jedoch auf der Erweiterung der Long-Short-term-Memory-Architektur (kurz: LSTM) zur Verbesserung des Informationserhaltes bei längeren Sätzen. Die genauen Vor- und Nachteile dieser Architektur sind im Kapitel 4.2.4 einsehbar.

Zusätzlich ist es bei twitter-hatespeech möglich ein Modell auf Basis von term frequency–inverse document frequency (kurz: TF-IDF) auszuwählen. [55] Bei TF-IDF handelt es sich nicht um ein DL-Modell, sondern um eine Funktion zur Wichtung von Wörtern mehrerer Sätze, die gemeinsam einen Korpus bilden. TF-IDF zählt nicht nur die Häufigkeit von Wörtern, sondern stellt diese mit der Häufigkeit des Wortes in anderen Bereichen des Korpus gegenüber. Dies ermöglicht die Verwendung von TF-IDF zur Bestimmung von Relevanz verschiedener Wörter für die Sätze. Eine Feature Extraction ist demnach ein geeigneter Anwendungszweck von TF-IDF. [56]

3.5 Social Media Plattformen

Welche Maßnahmen müssen soziale Medienplattformen gegen toxische Inhalte ergreifen? Im Jahr 2015 stieg die Zahl der Hassmeldungen gegen Menschen aufgrund ihrer Herkunft drastisch an. Google, Facebook und Twitter stimmten im selben Jahr zu, zukünftig Hassrede innerhalb von 24 Stunden zu entfernen. [57]

Folglich sind Soziale Netzwerke durch das NetzDG in Deutschland seit dem 1. Januar 2018 dazu verpflichtet, Hassrede und Falschmeldungen innerhalb von 24 Stunden zu entfernen. Diese Frist beginnt mit der Meldung des rechtswidrigen Inhalts durch andere Konten der Plattform. Andere rechtswidrige Inhalte müssen innerhalb von sieben Tagen nach Meldung an die Plattformen entfernt werden. [58]

Im Folgenden werden die Bemühungen, wenn möglich technische Details und Funktionen, gegen Toxizität und Hassrede der einzelnen Plattformen zusammengefasst.

3.5.1 Twitter

Entfernung von rechtswidrigen Inhalten

Twitter musste in Deutschland im April des Jahres 2017 insgesamt 50 Millionen Euro Strafzahlungen leisten, da die Fristen von 24 Stunden, beziehungsweise sieben Tagen, zu häufig nicht eingehalten wurden. [58] Im Dezember 2016 lag die Quote der bewerteten Meldungen innerhalb von 24 Stunden bei 19% aller Einsendungen. Diese Quote hat sich bis zum Mai 2017 auf 38% verdoppelt, liegt jedoch noch unter dem, von der EU-Kommission geforderten Wert, von 50%. [59]

Twitter nutzt automatisierte Systeme zur Erkennung verschiedener rechtswidriger Inhalte. Diese Systeme wirken regional, aber nicht für die gesamte Plattform. In Deutschland werden beispielsweise gemäß § 86a StGB Inhalte und Profile mit Hakenkreuzen blockiert. In anderen Ländern ist dies nicht der Fall, obwohl die Nutzungsbedingungen von Twitter

gewaltverherrlichende Gruppierungen, wie beispielsweise *"White Pride"*, verbieten. Twitter befindet sich in der Herausforderung zu evaluieren, wo in welchem Land die gesetzlich geschützte Meinungsfreiheit endet und wann ein strafvollzugsfähiger Tatbestand vorliegt. [60]

Twitter gab im März 2020 bekannt, dass die plattformeigenen Regeln zur Hassrede verschärft werden. [61] So sollen seither Inhalte verboten werden, die Menschen nach *"Rasse, Ethnie, oder nationaler Herkunft"* verurteilen. [62] Auf der anderen Seite steht Twitter in der Kritik, weil das Unternehmen bisher noch nicht hinreichend offengelegt, wie die Algorithmen zur Hasserkennung funktionieren und mit welchen genauen Kriterien das internationale Moderationsteam von Twitter arbeitet. [61]

Im Twitter NetzDG-Bericht Januar - Juni 2021 wird das Moderationsteam hinter den NetzDG-Beschwerden als über 150 Personen großes Team beschrieben, was speziell für die deutsche Gesetzesgebung geschult wurde. In einem dreiseitigen Text werden die verschiedenen Schulungswege und Kompetenzen des Teams erläutert. [63, S. 20–23]

In diesem Bericht befindet sich jedoch nicht die Rate der innerhalb von 24 Stunden entfernten gemeldeten Inhalte. Es ist aber die Metrik der gesamten Meldungen und der daraufhin gesperrten Inhalte enthalten. So wurden im ersten Halbjahr des Jahres 2021 insgesamt 833.402 Beiträge gemeldet, von welchen 81.320 gesperrt wurden. [63, S. 19] Dies entspricht 9,75% aller Meldungen. Im gleichen Zeitraum hat ein QS-Team etwa 5% der durch das Twitter-NetzDG-Team bearbeiteten Posts in einem Audit auf Korrektheit geprüft. [63, S. 22] Das Ergebnis dieses Audits ist nicht öffentlich einsehbar.

Safety Mode

Seit Anfang September 2021 verfügt Twitter über eine Funktion zur automatisierten Blockierung von Konten, die *"Tweets mit Beleidigungen oder hasserfüllte Bemerkungen senden"*. [64] Ob Konten blockiert werden, hängt auch von der Frequenz der Interaktionen zwischen dem Verfassenden und zu evaluierendem Nutzungskonto ab. Twitter ist sich der mentalen Risiken der Nutzung sozialer Plattformen bewusst und arbeitet mit Fachkräften aus der Psychologie an der neuen Funktion, um die Nutzungserfahrung auf Twitter angenehmer zu gestalten. [65]

Konten, denen gefolgt wird, erhalten keine automatische Blockierung. Für andere Konten kann global entschieden werden, ob diese einen, drei oder sieben Tage blockiert werden. Twitter selbst veröffentlicht nicht die technische Funktionsweise hinter dieser Funktion. Lediglich die Aktionen, die zu einer automatischen Blockierung führen, werden genannt. Zu denen zählt eine *"häufige oder aggressive Aktivität, die die Nutzungserfahrung stört"*. [66] Dazu zählt auch Spam in Form von *"häufigen und unaufgeforderten Erwähnungen mit unpassenden Hashtags zur Reichweitengenerierung"*. [66]

Künstliche Intelligenz

Um mit den immer wachsenden Datenmengen auf Twitter in der Kontrolle und Aussteuerung mithalten zu können, benötigt das Unternehmen Fachwissen im ML- oder DL-Bereich. Hierfür kaufte Twitter in der Vergangenheit Startup-Unternehmen mit dem nötigen Fachwissen. Durch die immer größere Wichtigkeit von Bildinhalten benötigt das Unternehmen zunehmend Expertise auf diesem Gebiet. Twitter nutzt beispielsweise seit dem Jahr 2018 eine KI für den Zuschnitt von Bildern in den Vorschauen. Ziel ist es, beim automatischen Zuschnitt keine interessanten Bildbereiche aus dem sichtbaren Rahmen zu entfernen. [67]

Bereits im Juli 2014 kaufte Twitter das Unternehmen *"Madbits"* für eine öffentlich unbekannte Geldmenge. Madbits spezialisierte sich auf eine automatische Erstellung von Bilddatenbanken. [68] Im Folgejahr 2015 kaufte Twitter mit *"Whetlab"* ein weiteres ML-Startup. Das zum Übernahmezeitpunkt 15 Monate alte Unternehmen spezialisierte sich in Big Data Analysis mit dem Schwerpunkt auf Mustererkennung in den Datensätzen. [69] Des Weiteren kaufte Twitter im Juni des Jahres 2016 das ML-Startup *"Magic Pony"*. [67] Dieses spezialisierte sich auf die Verbesserung unscharfer oder niederauflösender Bilder und Videos mittels KI. [70]

Mit dem Fortschreiten von Twitter im Bereich der KI passte die Plattform im Jahr 2017 den Feed der Startseite an. Ziel ist die Darstellung von Inhalten, die dem Nutzenden tendenziell am besten gefallen. Es werden NLP und Tweet-Ranking eingesetzt, um die Tweets in Reihenfolge kohärenter Präferenzen des angemeldeten Profils anzuzeigen. [67]

Twitter nutzt KI auch zur Erkennung von terroristischen Konten. So konnten innerhalb des ersten Halbjahres von 2017 bereits 300.000 terroristische Konten gesperrt werden. Im Jahr 2018 kündigte der damalige CEO Twitters Jack Dorsey an, dass die Plattform weiterhin KI zur Abwehr von Hassrede und Extremismus einsetzen werde. [67]

Dass entwickelte Algorithmen nicht immer eingesetzt werden, zeigt sich jedoch bei der Entfernung toxischer Inhalte im Bezug auf *"White-Supremacy"*. Ein von Twitter entwickelter Algorithmus zur Entfernung rechtswidriger Inhalte wurde für die Entfernung von ISIS-Propaganda eingesetzt, jedoch nicht für die Entfernung von White-Supremacy- und Neonazi-Inhalten. Eine bei Twitter angestellte Person erklärt, dass dieser Algorithmus berühmte republikanische Politiker und die Zitierung deren Äußerungen bannen würde, weshalb dieser Algorithmus nicht eingesetzt wird. Außerdem berge ein solcher Algorithmus eine hohe Falsch-Positiv-Rate und würde unschuldige Konten sperren. [71]

3.5.2 Facebook

Entfernung von rechtswidrigen Inhalten

In der Vergangenheit wurde Facebook häufig zu dem Umgang der Plattform mit Hassrede kritisiert. So untersagte Facebook zwar bis einschließlich 2017 Hass gegen Individuen, jedoch nicht gegen gesamte Gruppierungen von Menschen. Durch diese Regelung waren große Teile von Hassrede, besonders im Zusammenhang mit Rassismus und Antisemitismus, auf der Plattform durch die Allgemeinen Nutzungsbedingungen erlaubt und wurden nicht entfernt. [72]

In einer US-Studie zum Thema Hassrede im Zeitraum von zwei Monaten des Jahres 2021 schnitt Facebook mit TikTok auf dem letzten Platz bei der Entfernung antisemitischer Beiträge ab. Die Reichweite der 129 auf Facebook befindlichen Beiträge, die von dem Forschungsteam gemeldet wurden, betrug insgesamt 375.929 Impressionen. Facebook entfernte lediglich 5 der 129 (3,9%) Posts von der Plattform. Antisemitische Gruppen mit insgesamt 37.530 Mitgliedern wurden auch nach der Meldung an Facebook nicht entfernt. [73]

In den Facebook-Gemeinschaftsstandards zum Thema *“Hassrede”* befinden sich seit dem 16.12.2019 Regelungen, die sich gegen Antisemitismus richten. Verboten sind zu diesem Zeitpunkt immer noch lediglich direkte Beleidigungen, nicht die Verbreitung von Verunglimpfungen jüdischer Menschen, beispielsweise als *“heimliche Regierung”*. [74] Diese Aussagen sind erst nach fünf weiteren Revisionen der Standards von Facebook verboten, welche ab dem 11.08.2020 gültig sind. [75]

Was Facebook jedoch als Hassrede klassifiziert, ist nicht in jedem Fall strafrechtlich verfolgbar. Der Bundesgerichtshof entschied im Juli 2021, dass Facebook gelöschte Beiträge zweier Konten wieder freischalten muss. Grund hierfür ist die mangelnde strafrechtliche Relevanz der gesperrten Beiträge, obwohl diese nicht den Richtlinien von Facebook entsprechen. Laut Bundesgerichtshof müsse Facebook auch Konten über zukünftige Sperren informieren und anbieten, eine Stellungnahme zu den veröffentlichten Beiträgen abzugeben. [76] Facebook weiß um das Problem der Sperrung von Inhalten, die nicht gesperrt werden dürfen und bietet eine nachträgliche Freischaltung von gesperrten Inhalten routinemäßig seit Anfang 2019 an. [77]

Künstliche Intelligenz

Facebook gab im Mai des Jahres 2020 einen Leitplan für die Verwendungen von KI auf der Plattform bekannt. In diesem sind die bisherigen Bemühungen des Unternehmens zur KI im Kontext der Erkennung von Hassrede in einem Blogbeitrag zusammengefasst.

Im vierten Quartal des Jahres 2019 wurden 80,2% aller von Facebook als Hassrede klassifizierten und entfernten Inhalte auch von Facebooks KI als solche erkannt. Im ersten Quartal des Jahres 2020 liegt diese Rate bei bereits 88,8%. Im gleichen Zeitraum stieg die Anzahl der entfernten Inhalte mit durch Facebook klassifizierter Hassrede pro Quartal von 5,7 Millionen auf 9,6 Millionen an. [78]

Facebook nutzt für die NLP verschiedene Modelle. Unter anderem auch die von dem Forschungsteam von Facebook entwickelten XLM und XLM-R. [78] Bei diesen Modellen handelt es sich um Erweiterungen anderer Modelle, welche in den Kapiteln 5.2 und 5.3 erläutert werden.

Um die Anwendung dieser Modelle auf der Plattform in großem Maßstab zu ermöglichen, wurde ein Modell zur Simplifizierung von Inhalten auf Beitragsebene erstellt. So sollen Beiträge erkannt werden, die bereits in der Vergangenheit von dem Moderationsteam von der Plattform entfernt wurden. Dieses Modell soll eine Klassifizierung von Hassrede vereinfachen und wird laut Facebook aktiv eingesetzt. Facebook schreibt, dass diese Systeme niemals Perfektion erlangen werden, die KI jedoch weiterhin ein wichtiges Forschungsgebiet für die Erfüllung dieser Aufgaben ist. Komplexe und noch zu lösende Aufgaben stellen unter anderem die absichtliche Falschschreibung von Wörtern oder die Klassifizierung von (Text-)Inhalten dar, deren Bedeutung sich durch ein angehängtes Bild verändern kann. [78]

Die von Facebook eingesetzten Modelle verfügen seit dem zweiten Quartal des Jahres 2019 bereits eine Genauigkeit, die für den Livebetrieb der Plattform Entscheidungen treffen kann. Demnach wurden laut Facebook die meisten Meldungen von Hassrede vor der Einsendung durch Menschen bereits von den KI-Modellen als solche klassifiziert. Als Informationsquelle für die Modelle zählen die Texte, Bilder und Texte in Bildern von bereits entfernten und neuen Inhalten. [77]

Werden neue Inhalte von Facebooks KI-Modell als Hassrede klassifiziert und wurden gleiche oder ähnliche bereits von dem Moderationsteam gesperrt, entfernt das KI-Modell diese Beiträge seit dem 2. Quartal des Jahres 2019 ohne die Bestätigung eines Menschen zu benötigen. Kann dem Beitrag nicht automatisch einem früher entfernten Inhalt zugeordnet werden, muss ein Mensch die Entfernung autorisieren. Durch dieses System konnte Facebook die Quote der Erkennung von Hassrede vor der ersten Meldung aus der Community von 68% im ersten Quartal 2019 auf 80% im zweiten Quartal 2019 erhöhen. [77]

Die Errungenschaften gegen Hassrede auf Facebook sind auf die von Facebook erweiterten Modelle und neuartige Datensätze, wie den *“Hateful Memes Dataset”* zurückzuführen. [79] Die Genauigkeit der Modelle soll in Zukunft durch wenig- und selbstbeaufsichtigtes Lernen im Produktionsbetrieb verbessert werden. [78]

3.5.3 YouTube

Entfernung von rechtswidrigen Inhalten

Auf YouTube stehen verschiedene Werbeplätze zur Verfügung. Wird ein Video aufgerufen, wird eine sogenannte *"Pre-Roll-Ad"* vor dem ausgewählten Video gezeigt. Während der Wiedergabe eines Videos kann dieses von *"Mid-Roll-Ads"* unterbrochen werden. Neben dem Videoplayer auf der Seite befinden sich auf der Webseite und in den Apps auch verschiedene Orte für Bannerwerbungen. Somit steht die Werbung auf YouTube näher in Verbindung zu dem Inhalt eines Beitrags, als es bei anderen Plattformen der Fall ist.

Durch die Nähe der Werbung zum Inhalt der Videos stand YouTube in der Vergangenheit mehrfach in der Kritik von Werbetreibenden. Diese kritisieren die Platzierung der bezahlten Werbung vor, während oder nach Inhalten, die nicht mit den ethischen Standards der Werbetreibenden vereinbar sind.

Die Werbetreibenden sind die wichtigste Einnahmequelle für YouTube. Die Plattform erwirtschaftet schätzungsweise 15 Milliarden US-Dollar jährlich. Um diese Einkommensquelle zu erhalten, ergreift YouTube seit 2019 vermehrt monetäre Maßnahmen gegen Inhalte mit Pornographie, Hassrede, Beleidigungen und vulgärer Sprache. Diese Beiträge sind durch eine automatische Klassifizierung der Videoinhalte von der Monetarisierung auf der Plattform ausgeschlossen. [80]

Doch was ist mit den Inhalten, die sich nicht auf die Monetarisierung durch YouTube als Einkommensquelle verlassen müssen? Nicht nur neue Videos müssen auf der Plattform geprüft werden, sondern auch die bereits seit Jahren auf der Plattform befindlichen Videos. Eine Aufgabe, die enorme Rechenleistung für eine Plattform benötigt, auf welcher minütlich über 500 Stunden neues Videomaterial hochgeladen werden. [81]

Auf YouTube befinden sich öffentliche Inhalte, die den Holocaust leugnen oder stark gewaltverherrlichend sind, obwohl diese Beiträge gegen die Communityrichtlinien verstoßen. [80] Manche dieser Videos verfügen über 800.000 Aufrufe. Nach der Meldung von insgesamt 52 Videos und Konten durch ein US-Forschungsteam, wurden 5 Videos und 6 Konten von der Plattform entfernt. Mit einer Aktionsrate von 21,2% schneidet YouTube in dieser Studie besser als die konkurrierenden internationalen Plattformen ab. [73]

Nach eigener Aussage werden auf YouTube Inhalte entfernt, die etwa 0,17% der Reichweite der Plattform ausmachen. Bei einer Gesamtreichweite von 1 Milliarde gesehenen Stunden Videomaterial pro Tag, entspricht diese Metrik täglich etwa 1,7 Millionen Stunden betrachteten Inhalts, die gegen die Richtlinien von YouTube verstoßen. [82]

Auf YouTube werden jedoch nicht nur Videos hochgeladen. Unter den meisten Videos ist die Abgabe von Kommentaren erlaubt. Doch wie wird dieser Teil der Plattform moderiert?

Bei der Entfernung von Hasskommentaren unter Videos bezieht YouTube die Content-Creator mit ein. Erkennt YouTube bei dem Absenden eines Kommentars, dass es sich um Hassrede handeln könnte, wird dieser nicht für alle sichtbar veröffentlicht. Bevor ein solcher Kommentar veröffentlicht wird, muss dieser von dem Content-Creator freigegeben werden, unter dessen Video der Kommentar verfasst wurde. Wird ein Kommentar nach 60 Tagen nicht freigegeben, wird er automatisch gelöscht. [83]

Ferner soll auf YouTube bei dem Absenden eines Hasskommentars bereits auf die Natur des Kommentars aufmerksam gemacht werden. Wird ein solcher Kommentar abgesendet, blendet YouTube eine Meldung mit dem Hinweis auf die Communityrichtlinien ein. Der Kommentar kann nochmals geändert, oder ohne Änderung abgesendet werden. [7]

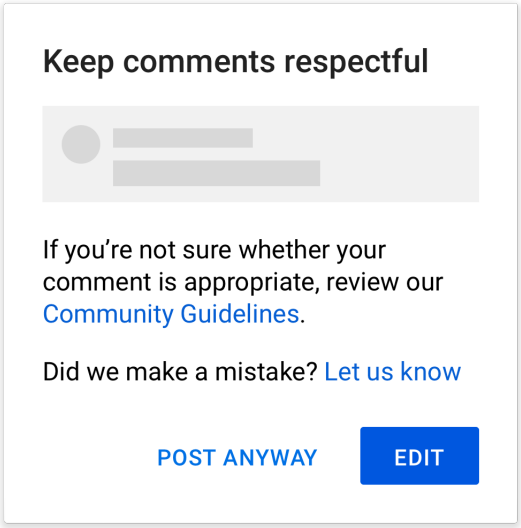

Abbildung 3.1: Warnung vor Absenden toxischer Kommentare auf YouTube [7]

Künstliche Intelligenz

Um die enormen Datenmengen der Plattform moderieren zu können, ist die Hilfe von KI notwendig. Durch das erhöhte Vorkommen von Beiträgen während der COVID-19-Pandemie, die gegen die Nutzungsbedingungen von YouTube verstoßen, setzt Google verstärkt KI zur Entlastung der Angestellten ein. [84]

Die KI-Modelle sperren seit 2020 vollautomatisch Inhalte, die Hassrede beinhalten und zu Gewalt aufrufen. Gehen Meldungen von nicht erkannten Inhalten bei YouTube ein, werden priorisiert die von *"Trusted Flaggern"* gemeldete Inhalte durch das Moderationsteam bearbeitet. Dies hat den Grund, dass die Genauigkeit dieser Meldungen durchschnittlich höher ist, als Meldungen aus der Community. Bei Trusted Flaggern

handelt es sich um ein *“Netzwerk von mehr als 180 Akademikern, Regierungsbehörden und Nichtregierungsorganisationen”*. [84]

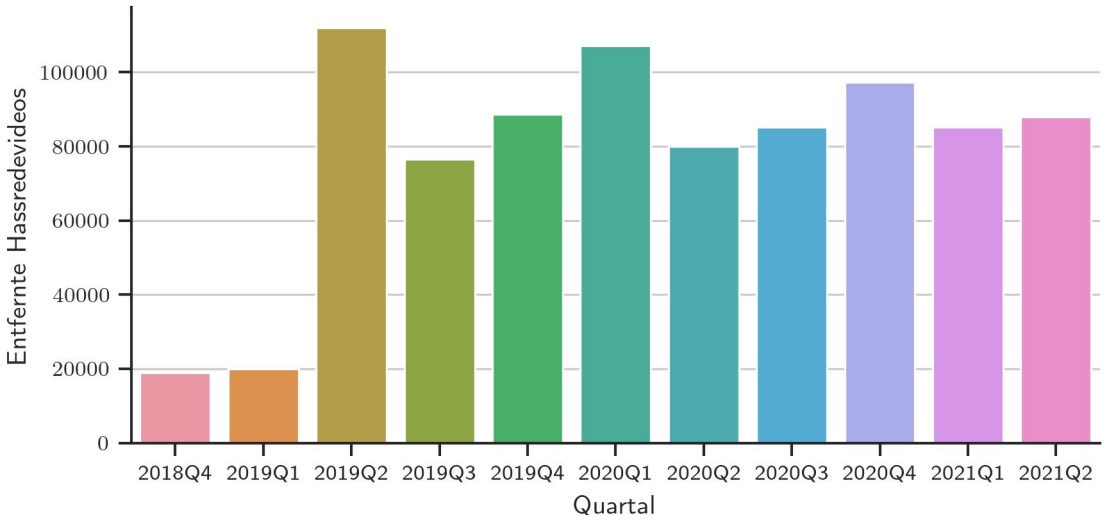

Abbildung 3.2: Von YouTube entfernte Hassredevideos nach Quartal (Abbildung nach [84])

Google schreibt nicht welches Modell für die Erkennung von Hasskommentaren auf YouTube verwendet wird und auch nicht wie das System funktioniert. Google ist jedoch sehr aktiv in der KI-Entwicklung und dokumentiert Modelle, die für solche Aufgaben geeignet sind. Im Kapitel 5.1 wird die BERT-Architektur erklärt, die auch für die Google Jigsaw Perspective API ein wichtiges Forschungsergebnis darstellt.

3.6 Zusammenfassung

Toxische Inhalte sind ein Problem für viele Plattformen sozialer Medien. Zum einen sind die Plattformbetreibenden international an rechtliche Vorgaben gebunden und müssen zeitnah Inhalte entfernen. Zum anderen möchten sich die Plattformen absichern und entfernen somit automatisiert Inhalte. Aus diesem Grund und den plattformeigenen Richtlinien, die zum Teil mehr verbieten als rechtlich in einzelnen Ländern notwendig ist, können Gerichte die Freischaltung von gesperrten Inhalten anordnen.

Um die Compliance der Unternehmen zu verbessern, sind immer besser werdende KI-Modelle gefragt. Wichtig sind bei der Bewertung der Inhalte demnach nicht nur die Richtig-Positiv-Rate, sondern auch die Richtig-Negativ-Rate.

4 Neuronale Netze

In dieser Masterthesis soll DL eingesetzt werden, um die NLP-Aufgabe der Toxizitätserkennung zu bewältigen. In diesem Kapitel werden die Formen und Funktionsweisen von künstlichen neuronalen Netzen erläutert.

Neuronale Netze sind der Grundstein eines DL-Modells. Sie sind auch unter den Namen Artificial Neural Network (kurz: ANN) oder Simulated Neural Network (kurz: SNN) bekannt. Der Aufbau und die Funktionsweise eines ANNs ist dem menschlichen Gehirn nachempfunden. [85, S. 23] ANNs sind für die Automatisierung von Aufgaben der Mustererkennung geeignet. [86, S. 1]

4.1 Aufbau

Um die Eigenschaften und Unterschiede der einzelnen Modelle betrachten zu können, muss zunächst der allgemeine Aufbau eines ANNs erläutert werden.

4.1.1 Das künstliche Neuron

Ein ANN besteht aus vielen untereinander vernetzten Recheneinheiten, die auch (künstliche) Neuronen genannt werden. Diese Neuronen können eine zuvor definierte Anzahl von Eingangssignalen zu einem Ausgabewert verarbeiten. [86, S. 1]

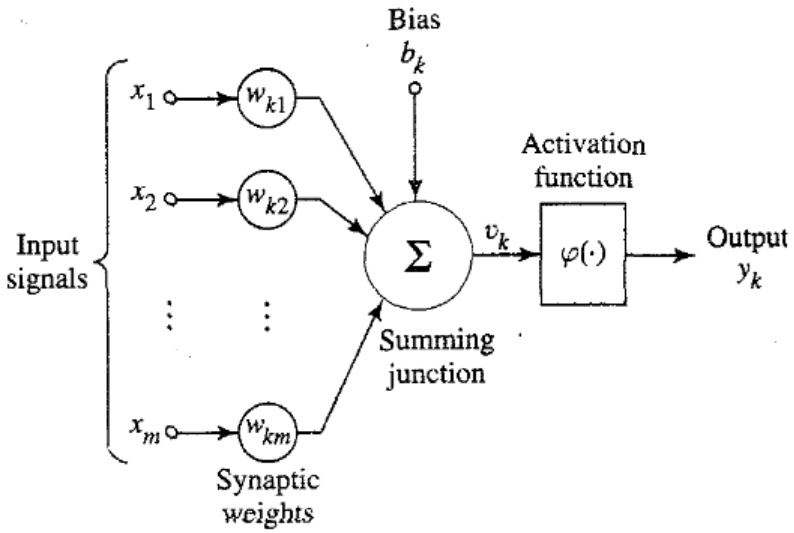

Abbildung 4.1: Aufbau eines künstlichen Neurons [85, S. 33]

Jedes Neuron empfängt Zahlen, verarbeitet diese weiter und gibt anhand der Aktivierungsfunktion eine Ausgabe zurück. Jedem Eingangswert $[x_1, x_{...}, x_m]$ wird eine Wichtung

$[W_{x_1}, W_{x_{...}}, W_{x_m}]$ zugeordnet. Anschließend werden die gewichteten Eingangswerte addiert. Die Summe kann optional ein Bias b_k zur Verschiebung des Wertes erhalten, bevor diese als Wert υ_k an die Aktivierungsfunktion übermittelt wird. [85, S. 33–34]

$$\upsilon_k = \sum_{j=1}^{m} x_j * W_{k_j}$$

Eingangswert für die Aktivierungsfunktion des Neurons nach [85, S. 33]

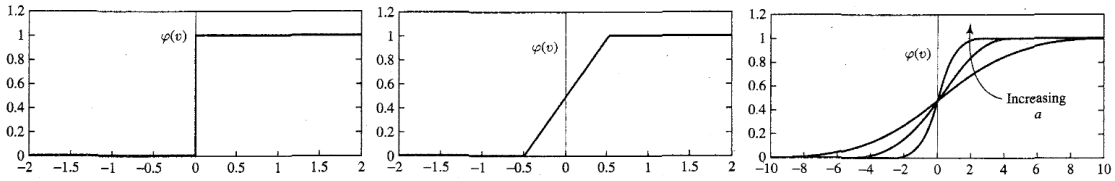

Abbildung 4.2: Aktivierungsfunktionen künstlicher Neuronen [85, S. 35]

Ein künstliches Neuron kann mithilfe einer definierten Funktion φ aktiviert werden. Mögliche Funktionen sind beispielsweise Schwellenwerte, stückweise lineare Funktionen oder Sigmoide. [85, S. 34–36]

$$\varphi(\upsilon) = \begin{cases} 1, & \upsilon_k \geq 0 \\ 0, & \upsilon_k < 0 \end{cases}$$

Schwellenwertfunktion bei Null nach [85, S. 34]

$$\varphi(\upsilon) = \begin{cases} 1, & \upsilon \geq +\frac{1}{2} \\ & \upsilon, +\frac{1}{2} > \upsilon > -\frac{1}{2} \\ 0, & \upsilon \leq -\frac{1}{2} \end{cases}$$

stückweise lineare Funktion nach [85, S. 36]

Laut Haykin ist der Sigmoid die häufigste Darstellung einer Aktivierungsfunktion künstlicher Neuronen. Durch die Anpassung des Wertes a kann die Steigung des Sigmoiden angepasst werden. [85, S. 36] Somit reicht die Änderung eines einzigen Wertes der künstlichen Neurone um das Ergebnis der Aktivierungsfunktion maßgeblich zu beeinflussen. Die weiteren trainierbaren Parameter sind die Wichtungen der Eingangsdaten und der Bias der summierten gewichteten Eingangsdaten.

$$\varphi(\upsilon) = \frac{1}{1 + \exp(-a\upsilon)}$$

Sigmoidfunktion nach [85, S. 36]

Sind alle Bestandteile des künstlichen Neurons definiert, kann eine Ausgabe y_k berechnet werden. Für diese Ausgabe wird die gewichtete Summe der Eingangswerte v_k mit dem Bias b_k addiert und an die Aktivierungsfunktion φ übergeben.

$$y_k = \varphi(u_k + b_k)$$

Ausgabe des Neurons nach [85, S. 33]

4.1.2 Input-Layer

Die Neuronen des Input-Layer werden als Input-Neuronen bezeichnet. Input-Neuronen führen keine Berechnungen durch, sondern stellen lediglich eine Darstellung der Eingangsdaten dar. Jedes ANN muss über einen Input-Layer verfügen. [85, S. 43] Soll beispielsweise ein Wort klassifiziert werden, welches vorher mit einem 50-dimensionalen word2vec-Modell in einen Vektor umgewandelt wurde, werden dementsprechend 50 Input-Neuronen benötigt um jede Dimension des Vektors zu verarbeiten.

4.1.3 Hidden-Layer

Die verarbeitenden Schichten, die nicht die Ausgabeschicht sind, werden Hidden-Layer genannt. Die Hidden-Layer werden als versteckt bezeichnet, da diese weder im Input noch im Output über direkten Kontakt zu ihnen verfügen. Ein ANN kann über beliebig viele Hidden-Layer verfügen und sogar gänzlich auf diese verzichten. Jeder Hidden-Layer kann über beliebig viele Neuronen verfügen. Die Anzahl der Neuronen in den verschiedenen Hidden-Layer eines Modells muss nicht identisch sein. Die Art eines neuronalen Netzes bestimmt die Verbindungen zwischen den Hidden-Layer. [85, S. 43]

4.1.4 Output-Layer

Die Neuronen des Output-Layer werden als Output-Neuronen bezeichnet. Output-Neuronen sind die letzte Ebene der Datenverarbeitung und geben das Ergebnis des ANNs aus. Jedes ANN muss über einen Output-Layer verfügen. Die Anzahl der Output-Neuronen entspricht in vielen Modellarchitekturen der Anzahl der unterschiedlichen Klassen. [86, S. 4] Soll beispielsweise der Klassifikator handgeschriebene Zahlen von 0 bis 9 erkennen, benötigt ein Modell zehn Output-Neuronen.

4.2 Arten neuronaler Netze

Die Funktion und das mögliche Anwendungsgebiet eines ANNs wird maßgeblich durch die Grundarchitektur des Modells bestimmt. Welche Architekturen bisher zur Verfügung stehen und was deren häufig genutztes Anwendungsgebiet ist, wird in den folgenden Unterpunkten erläutert.

4.2.1 Single-Layer Feedforward Neural Network

Ein Single-Layer Feedforward Neural Network (kurz: SLFN) ist ein aus einer Verarbeitungsschicht bestehendes ANN. So verfügt ein SLFN lediglich über einen Input- und einen Output-Layer. Der Input-Layer zählt nicht zu den Ebenen dazu, da in dieser Ebene keine Rechenoperationen durchgeführt werden. SLFNs werden für simple Klassifizierungen mit wenigen möglichen Klassen verwendet. [85, S. 43]

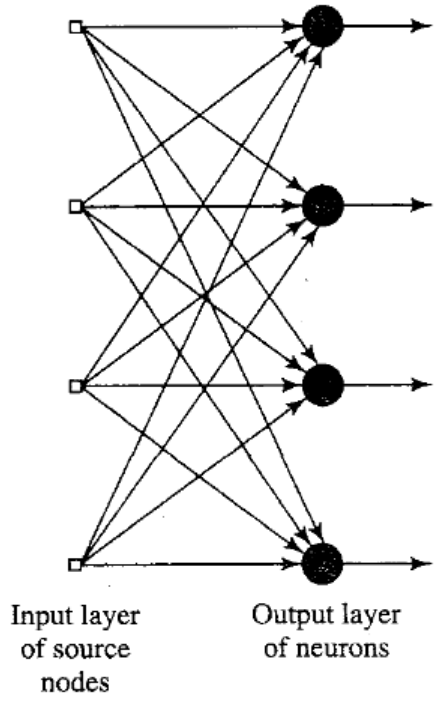

Abbildung 4.3: Aufbau eines SLFNs mit 4 Neuronen [85, S. 43]

Bei einem Feedforward Neural Network (kurz: FNN) ist jede Ausgabe eines Neurons einer Ebene mit jedem Eingsangswert eines Neurons der nächsten Ebene verbunden. Die Neuronen innerhalb einer Ebene sind nicht miteinander verbunden. [85, S. 43] Aus diesem Grund beträgt die Verbindungsanzahl eines SLFNs die Anzahl der Input-Neuronen multipliziert mit der Anzahl der Output-Neuronen.

4.2.2 Multilayer Feedforward Neural Network

Um komplexere Aufgaben lösen zu können, werden mehr als eine Verarbeitungsschicht benötigt. Verfügt ein ANN über mehrere Hidden-Layer, wird es auch als tiefes ANN (Deep Neural Network) bezeichnet. Handelt es sich bei diesem um ein FNN, ist es ein Multilayer Feedforward Neural Network (kurz: MLFN). Im Rahmen dieser Masterthesis verfügt jedes neuronale Netz über mehrere Hidden-Layer. Aus diesem Grund handelt es sich bei all diesen ANNs um tiefe neuronale Netze und somit um DL-Modelle. [87]

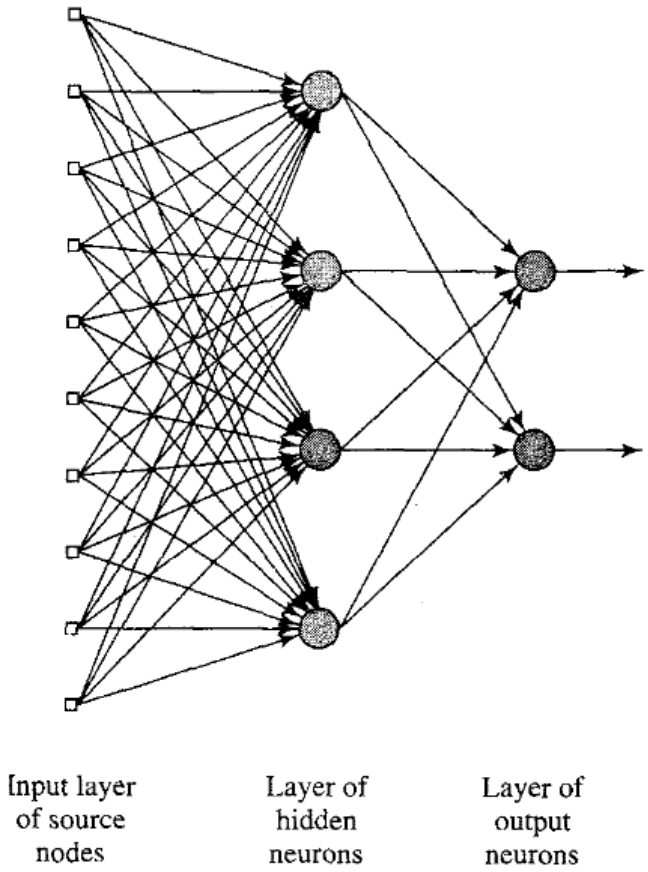

Abbildung 4.4: Aufbau eines MLFNs mit 10 Eingangsneuronen, 4 Verarbeitungsneuronen in einem Hidden-Layer und 2 Ausgangsneuronen in einem Output-Layer [85, S. 44]

4.2.3 Recurrent Neural Networks

Ein RNN ist ein FNN, welches über mindestens eine rückführende Schleife verfügt. Es kann als Single-Layer- oder Multilayer-Modell dargestellt werden. [85, S. 45]

RNNs werden für Aufgaben verwendet, bei denen eine einzige Reihe von Eingangsdaten zusammenhängend bewertet werden sollen. So kann beispielsweise nicht nur ein einziger Wortvektor klassifiziert werden, sondern eine zusammenhängende Folge dieser. Durch die rückführende Schleife werden die Informationen des vorherigen Wortvektors in die nächste Iteration übertragen. Bei diesem Prozess erhalten die Neuronen

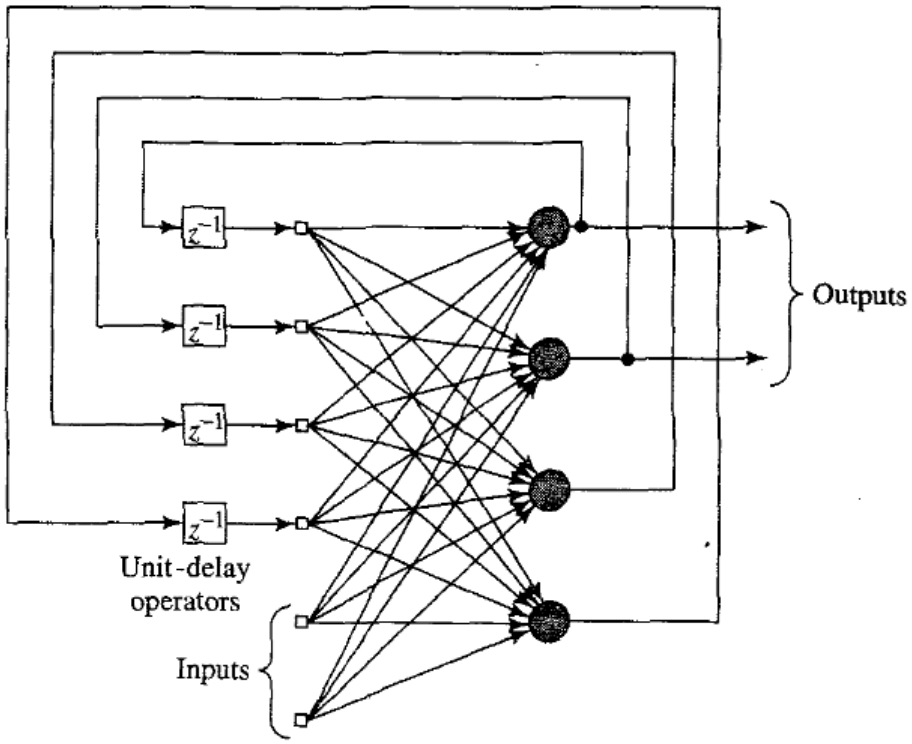

Abbildung 4.5: Aufbau eines RNNs mit 4 Neuronen und einem Hidden-Layer [85, S. 46]

zu einem gewissen Anteil die neuen Eingangsdaten und zu einem gewissen Anteil die bereits verarbeiteten Daten. Das führt zu dem Problem, dass neue und alte Daten nicht gleichmäßig repräsentiert werden. Dieses ist das Problem der *"Vanishing Gradients"*. [88]

Informationsanteil/ Eingangsdaten	1	2	3	4	5	6	7
1	100%						
2	50%	50%					
3	25%	25%	50%				
4	12,5%	12,5%	25%	50%			
5	6,25%	6,25%	12,5%	25%	50%		
6	3,125%	3,125%	6,25%	12,5%	25%	50%	
7	0,78%	0,78%	3,125%	6,25%	12,5%	25%	50%

Tabelle 4.1: RNN und Informationsgehalt einzelner Eingangsdaten mit einer Wichtung neuer Vektoren zu 50%

Wird in einem Beispiel von einer Aufteilung der Arbeitsschritte von 50% neuer und 50% bereits verarbeiteter Daten ausgegangen, können lediglich zwei Wortvektoren gleichwertig dargestellt werden. Kommt ein dritter Wortvektor dazu, wird dieser erneut mit 50% gewichtet, während die vorherigen Daten sich die verbleibenden 50% teilen. Mit jedem weiteren Wortvektor folgt eine weitere Iteration. Somit beinhaltet das ANN kaum noch Informationen der Wortvektoren vom Verarbeitungsbeginn.

Der Effekt der Vanishing Gradients kann mit einer anderen Eingangsdatenwichtung als 50% minimiert werden. Um die Abweichung der Informationsgehalte einzelner Daten im RNN gering zu halten, kann beispielsweise die Wichtung so gewählt werden, dass der erste Datensatz etwa der Wichtung des letzten Datensatzes entspricht. Bei einer Anzahl von sieben zusammenhängenden Eingangsdaten liegt die Eingangswichtung für eine gleichmäßigere Verteilung bei etwa 22% und die Wichtung der bereits verarbeiteten Daten dementsprechend bei etwa 78%, wie der Tabelle 4.2 zu entnehmen ist.

Informationsanteil/ Eingangswichtung	1	2	3	4	5	6	7
10%	53,1%	5,9%	6,6%	7,3%	8,1%	9,0%	10%
20%	26,2%	6,6%	8,2%	10,2%	12,8%	16,0%	20%
22%	22,5%	6,4%	8,1%	10,4%	13,4%	17,2%	22%
25%	17,8%	5,9%	7,9%	10,6%	14,1%	18,8%	25%
30%	11,8%	5,0%	7,2%	10,3%	14,7%	21,0%	30%

Tabelle 4.2: RNN und Informationsgehalt einzelner Eingangsdaten mit unterschiedlichen Eingangswichtungen

Mit gut erhaltenen Informationen einer Sequenzlänge von Sieben können jedoch nicht alle Aufgabenstellungen bewältigt werden. Die Klassifizierung toxischer Kommentare im Internet benötigt für die meisten Kommentare deutlich längere Sequenzen. Wie die Sequenzlänge sich auf die Genauigkeitswerte eines Modells auswirkt, wird in der Modelloptimierung dieser Masterthesis beschrieben.

4.2.4 Long Short-term Memory

Während mithilfe der Eingangswichtungen die Informationen zurückliegender Daten der Sequenz bei kurzen Sequenzen gut erhalten werden können, benötigen längere Sequenzen andere Lösungsansätze. Um das Problem der Vanishing Gradients zu lösen, wurde im Jahr 1997 die bis heute relevante LSTM-Architektur von Hochreiter und Schmidhuber vorgestellt. [88, 89]

LSTM stellt eine Erweiterung der RNN-Architektur dar. Eine langfristige Speicherung von Informationen innerhalb des ANNs wird über sogenannte *“Forget-Gates”* realisiert. [89] Diese sind neben dem Input und dem Output einer Ebene für die Aufbewahrung von Daten zuständig. Forget-Gates entscheiden anhand einer Sigmoidfunktion über die Wichtigkeit eines Wertes. Anhand der Wichtigkeit der Information wird die Anteil dieser Information im gesamten Netz definiert. [88]

Während LSTM die RNN-Architektur in der Genauigkeit aufwertet, fügt es ist die Ausführungszeit und Trainingszeit verglichen zu anderen Architekturen langsam und ressourcenintensiv. [90, S. 2]

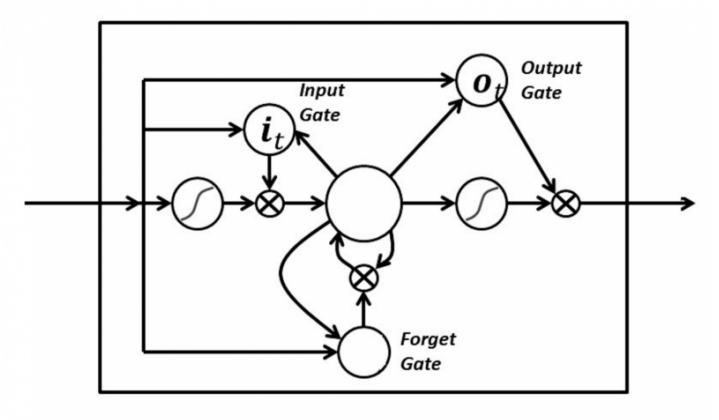

Abbildung 4.6: Erweiterung der RNN-Architektur um ein Forget-Gate [88]

4.2.5 Transformer-Architektur

Die Transformer-Architektur wurde mit dem Paper *"Attention Is All You Need"* von Vaswani et al. 2017 beschrieben. Die Mitwirkenden waren zum Forschungszeitpunkt in den Arbeitsgruppen Google Brain und Google Research tätig. Mit dieser Veröffentlichung wurde der Grundstein für die neuen SOTA-Modelle gelegt, indem es die Ausführungsgeschwindigkeit, Trainingskosten und die Genauigkeit gegenüber bisheriger Modelle übertrifft. [90]

Da die Transformer-Architektur auf rückführende Schleifen verzichtet, ist sie besonders gegenüber RNN- und LSTM-Architekturen in der Ausführungszeit überlegen. Bei RNN und LSTM muss jedes Datum eines zu klassifizierenden Datensatzes nacheinander eingelesen werden, ohne die Möglichkeit diese Prozessinstanz zu parallelisieren. Moderne Rechentechnik macht besonders in der Parallelisierung erhebliche Fortschritte, weshalb dieser Punkt für eine skalierbare Anwendung eine kritische Rolle spielt. Das von Vaswani et al. 2017 trainierte Transformer-Modell konnte bereits nach zwölf Stunden mit acht NVidia P100 GPUs in der Übersetzung zwischen Englisch, Deutsch und Französisch eine höhere Genauigkeit als alle vorherigen ANNs erreichten. [90, S. 7]

Was macht ein Transformer-Modell anders als die bisher genutzten Modelle, um diese Ergebnisse zu erzielen? In Abbildung 4.7 ist der Aufbau der Transformer-Architektur dargestellt. Ein Transformer-Modell kann über eine beliebige Anzahl von Encodern verfügen. Jeder Encoder kann eine beliebige Anzahl von Hidden-Layer und Neuronen beinhalten. Wird ein Input-Embedding eingelesen, so wird diesem ein Positional-Encoding anhand der Position in der Sequenz addiert. Hierdurch wird dem Wort ein Kontext zugeordnet. [90, S. 6]

Um die Sprache besser in einem Modell abbilden zu können, müssen die Beziehungen unter den Wörtern einer Sequenz quantifiziert werden. In der Transformer-Architektur wird dies über *"Attention"* realisiert. Attention beschreibt die Wichtigkeit, über die andere

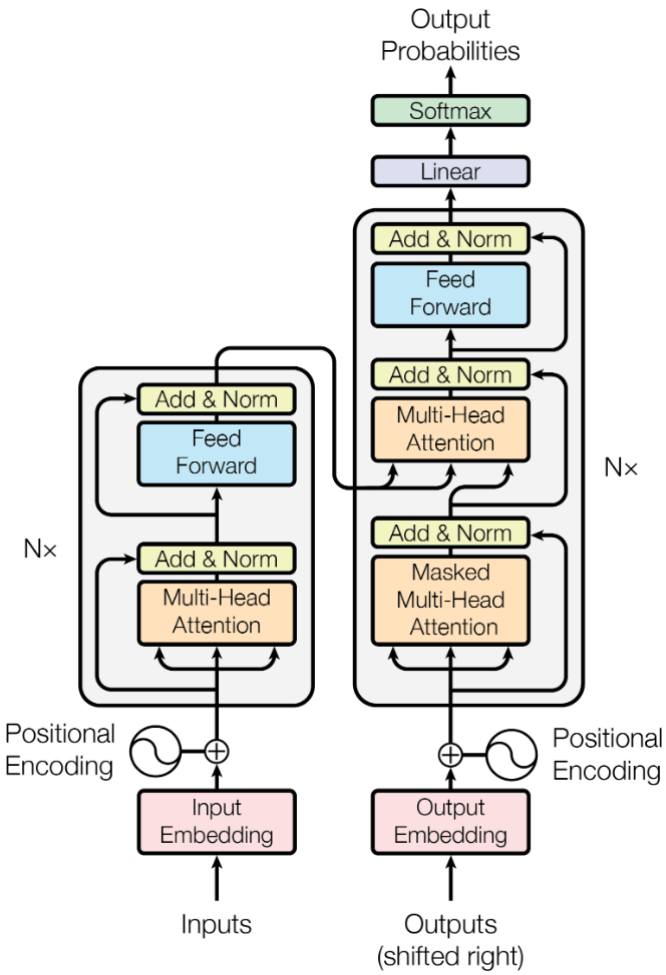

Abbildung 4.7: Die Transformer-Architektur [90, S. 3]

Wörter zu dem betrachteten Word verfügen. Somit lässt sich zuordnen, welche Artikel, Verben oder Adjektive zu welchen Substantiven gehören. Der Attention-Vektor eines Wortes verfügt über so viele Wichtigkeiten im Intervall $[0, 1]$ wie die Sequenz lang ist, wodurch jedem Wort eine Wichtigkeit zu einem anderen oder zum selben Wort zugeordnet wird. [90, S. 3]

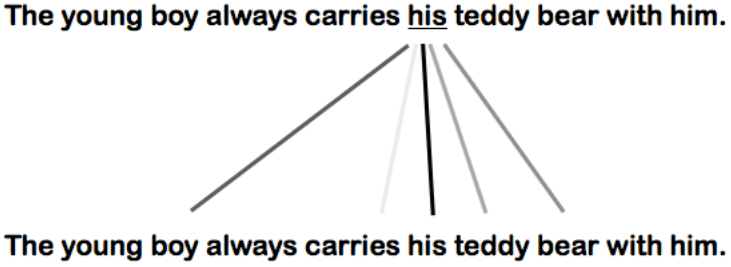

Abbildung 4.8: Beispiel der Attention des Wortes *"his"* im Beispielsatz [91]

Jedes Wort bewertet sich selbst als sehr wichtig und hat damit eine hohe Attention zu sich selbst im eigenen Attention-Vektor. Dieses Verhalten wird mittels der Multi-Head-Attention

normalisiert. Die Multi-Head-Attention rechnet die gewichtete Attention aus h parallelen Attention-Ebenen aus. Im Paper *"Attention Is All You Need"* wurde $h = 8$ gewählt. Mit dieser Methode verfügt jedes Wort nur noch über lediglich einen Attention-Vektor. [90, S. 4–5]

Die Attention-Vektoren werden in einem FNN weiterverarbeitet. Dabei sind die Verarbeitungsschritte voneinander unabhängig, wodurch eine Parallelisierung dieses Prozesses stattfinden kann. Bei der Verarbeitung der einzelnen Wörter werden die in der Sequenz folgenden Wörter mit Nullen gefüllt. Dies hat den Grund, dass das ANN das jeweils nächste Wort erraten soll und somit ein Lerneffekt entsteht. Dieser Prozess wird so lange wiederholt, bis ein *"End of Sentence Token"* (kurz: `<EOS>`) generiert wird. Ist die zu verarbeitende Sequenz kürzer als die vom Modell zugelassene Länge, wird mit einem `<pad>` aufgefüllt. Weiterhin werden *"Classification"*-Token (kurz: `<CLS>`) zur Darstellung der Sequenzlänge verwendet. [90, S. 13]

Die Transformer-Architektur ist seit diesem Zeitpunkt in fast allen Modellen vertreten, die im SuperGLUE-Leaderboard auftauchen. [92, S. 8–9] Der *"General Language Understanding Evaluation"* (kurz: GLUE) ist ein Standard zur Genauigkeitsmessung von Modellen in NLP-Aufgaben. [92, S. 1–2] Google nutzt die Fortschritte der Transformer-Architektur, um die Suchanfragen auf der Plattform besser zu verstehen. Für das Unternehmen ist die Transformer-Architektur *"eines der größten Errungenschaften der Geschichte für Suchmaschinen"*. [93]

Wie Google und andere Plattformen Transformer für NLP-Aufgaben nutzen und inwiefern diese Modelle die Architektur erweitern, wird im Kapitel *"5 Anpassungen der Architekturen"*

4.3 Lernmethoden

Wie kann Rechenleistung für die Genauigkeitsverbesserung eines ANNs verwendet werden? Die Stellschrauben für das ANN sind die Eingangswichtungen und Biases der einzelnen Neuronen. Diese zufällig zu ändern und dadurch auf eine bessere Genauigkeit zu erzielen, würde im Trainingsprozess eines ANNs zu viel Rechenzeit benötigen, um damit wirtschaftlich Modelle erstellen zu können. Stattdessen wird ein Prozess namens *"Backpropagation"* durchgeführt, mit dessen Hilfe die beste lokale Lösung des ANNs gefunden wird. [94]

Mit Backpropagation wird der schnellstmögliche Trainingsschritt gesucht, der die größtmögliche Verbesserung der Modellgenauigkeit verursacht. Hierfür wird der Gradientenabstieg berechnet, welcher mit der Backpropagation im Netzwerk umgesetzt wird. [94] Um die Backpropagation durch die Reduktion von Rechenschritten so effektiv wie möglich zu gestalten und die Lernfähigkeit des Netzwerks über lineare Zusammenhänge hinaus zu

erhalten, sind Aktivierungsfunktionen künstlicher Neuronen zumeist Sigmoidenfunktionen im Intervall $[0,1]$ oder $[-1,1]$. [94] [85, S. 36 & 183–186]

Um den Fortschritt I des Lernprozesses messbar zu machen, wird der Mean Squared Error (kurz: MSE) zwischen den generierten Label y' und den tatsächlichen Klassen y gebildet und der Mittelwert aller ausgerechneten MSE im Trainingsdatensatz m gebildet. Dies beschreibt die *"Cost"* eines Netzes und steht dafür wie schlecht die Klassifizierung des ANNs ist. Für die Berechnung des Cost-Wertes sind auch die Verwendung des Mean Errors, Mean Absolute Errors und Root Mean Squared Errors verbreitet. [94]

$$I = \sum_{i=1}^{m} \frac{1}{m}(y - y')^2$$

Cost-Funktion nach [94]

Die Cost-Funktion kann um eine *"Loss"*-Funktion erweitert werden. In einer Loss-Funktion können zusätzliche Metriken berücksichtigt werden, die nicht von der Cost-Funktion abgebildet werden. Mit der Senkung des Cost- oder Loss-Wertes wird der Lernfortschritt gemessen. [94]

Als Ursprungswerte für ein ANN werden zunächst Zufallszahlen für alle Wichtungen und Biases generiert. Dementsprechend erscheint das Verarbeitungsergebnis des ANNs zufällig und gibt keinerlei verwertbare Funktion zurück. Die Backpropagation passt die Klassen einzeln an, um den Cost-Wert zu verringern. Hierbei werden erst die Klassen betrachtet, deren Anpassung den höchsten Genauigkeitsgewinn bringen. Ausgehend von dem Outputneuron werden die Eingangswichtungen betrachtet. Die Backpropagation besitzt die Information der richtigen Klassifizierung und kann Schritt für Schritt die fehlleitenden Eingangswichtungen erhöhen oder verringern. Je näher der Cost-Wert an die Null annähert, desto geringer fallen die Anpassungen der Wichtungen durch die Backpropagation aus. Die von der Backpropagation ausgeführten Änderungen werden von der *"Learning Rate"* multipliziert mit dem Ergebnis des Gradientenabstiegs bestimmt. [95]

Doch mit welchen Lernmethoden können ANNs eine Klassifizierung oder die Erfüllung anderer Aufgaben beigebracht werden? In den folgenden Kapiteln werden die Methoden und Anwendungszwecke dieser erläutert.

4.3.1 Supervised-Learning

Mit Supervised-Learning wird der Prozess des Anlernens eines Modells mittels bereits durch Menschen annotierter Daten beschrieben. Mithilfe der annotierten Daten kann im Lernprozess der Cost- und Loss-Wert berechnet und somit der Lernfortschritt messbar

gemacht werden. Durch dieses Wissen kann der Lernprozess von Algorithmen, wie der Backpropagation, betreut werden. Aus diesem Grund spricht man vom beaufsichtigten Lernen. Ein Nachteil des Supervised-Learnings ist der hohe Rechenzeitbedarf während des Trainingsprozesses. [96]

Supervised-Learning kann für die Klassifizierung von Daten in feste Kategorien eingesetzt werden. Diese können binär sein, wie beispielsweise ein Spam-Filter, oder auch zwischen vielen Klassen unterscheiden, wie eine Objekterkennung mittels optischer Sensorsdaten. [96]

Ferner kann mit Supervised-Learning eine Regression trainiert werden. Mit einer Regression kann mit einer Vielzahl von Eingangsdaten eine Vorhersage für bestimmte Kennwerte umgesetzt werden. Regressionsaufgaben werden unter anderem mit linearer, logistischer oder polynomialer Regression gelöst. [96]

4.3.2 Unsupervised-Learning

Unsupervised-Learning beschreibt das Training eines ML-Algorithmus, ohne des Vorhandenseins durch Menschen annotierter Daten. Aus diesem Grund wird dieser Prozess unbeaufsichtigtes Lernen genannt. Der Nachteil dieser Methode ist die Ungenauigkeit der Ergebnisse, wenn diese nicht von Menschen geprüft und angepasst werden. [96]

Unsupervised-Learning wird unter anderem für die Erkennung von Zusammenhängen in Datensätzen verwendet. Diese spielt beispielsweise in der Gruppenerkennung und der Präferenzerkennung eine Rolle. Des Weiteren kann eine Dimensionsverringerung mit Unsupervised-Learning umgesetzt werden. [96]

Des Weiteren kann ein Modell mit einer Mischform aus Un- und Supervised-Learning trainiert werden. Dieser Prozess heißt *“Semi-Supervised-Learning”*. Anwendungsgebiete für diese Methode sind unter anderem jene Aufgabenstellungen, bei denen die annotierten Trainingsdaten nicht für eine ausreichende Genauigkeit des Modells ausreichen. [97, S. 41]

4.3.3 Reinforcement Learning

Was ist, wenn komplexere Zusammenhänge in einem DL-Modell abgebildet werden sollen, wie beispielsweise die eigenständige Bewegung einer Maschine oder das Lernen eines Spiels? Für diese Aufgaben ist das Reinforcement Learning konzipiert. Für das Reinforcement Learning werden, je nach Aufgabe, keine Datensätze benötigt. [98]

Das Modell lernt durch Trial-and-Error-Sequenzen eines sogenannten *“Agenten”*. Um dem Agenten einen Fortschritt zu kommunizieren, wird eine Belohnung definiert, die es durch das Modell zu maximieren gilt. Belohnungen können durch nicht gewollte Aktionen dem Agenten auch wieder entzogen werden. Der Agent beginnt zunächst mit zufälligen Aktionen und wiederholt bevorzugt die Vorgänge, die den Belohnungsstand am schnellsten erhöhen. [98]

Mit genügend Iterationen kann dem Modell somit ein komplexer Zusammenhang, wie beispielsweise das Spielen eines Videospiels oder das Fahren eines Autos, beigebracht werden. Tesla beispielsweise nutzt eine Reihe von ANNs, um den Modellen das autonome Fahren beizubringen. So generiert Tesla aus realen Aufnahmen Videospielsimulationen, die wiederum mit einer weiteren KI realitätsnah dargestellt werden. Aus diesen realitätsnah gerenderten 3D-Simulationen wird mithilfe von Reinforcement Learning das Modell hinter dem autonomen Fahren trainiert. [99]

4.3.4 Generative Adversarial Networks

Ein *“Generative Adversarial Network”* (kurz: GAN) ist eine Lernmethode von ANNs und gilt als Unterkategorie des Semi-Supervised-Learnings. Mithilfe von GANs erhalten Modelle ein tiefgreifendes Verständnis des zu lernenden Sachverhalts, mit welchem eine Generierung neuer Inhalte möglich ist. [100] GANs zählen laut OpenAI zu den vielversprechendsten Methoden, um Wissen darzustellen. [101]

Doch wie funktioniert diese Trainingsmethode? Ein GAN benötigt für das Training zwei konkurrierende Modelle: einen *“Generator”* und einen *“Discriminator”*. Die Aufgabe des Generators ist die Erstellung neuer Inhalte, die den Trainingsdaten ähneln. Der Discriminator muss zwischen echten und generierten Inhalten entscheiden. Somit wird die Qualität des Generators gemessen und kann gezielt durch Backpropagation trainiert werden. Während dieses Trainingsprozesses lernt der Generator immer besser nachzuahmen und der Discriminator immer besser dies zu erkennen. [101]

Mit dem trainierten GAN können auf Basis des Trainingsdatensatzes komplett neue Daten generiert werden. Hierfür wird dem Generator ein Rauschen zugeführt. [100] Ist ein Modell korrekt trainiert, wandelt es das Rauschen in neue Daten um, die nicht dem Trainingsdatensatz entsprechen, aber dieser Grundstruktur entsprechen. Auf diese Weise funktioniert die Webseite `this-person-does-not-exist.com`, welche alle zwei bis drei Sekunden neue Gesichter von nicht existierenden Menschen generiert. Diese Gesichter sind so hochauflösend und menschennah, dass diese durch Menschen selten von echten Fotos zu unterscheiden sind. [102]

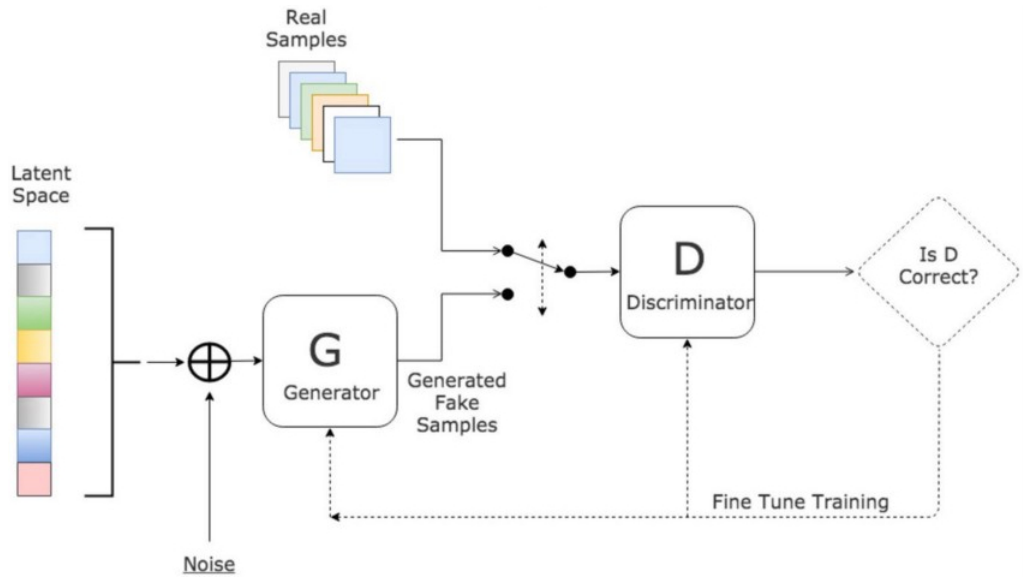

Abbildung 4.9: Aufbau eines GANs [100]

4.4 Kennzahlen

ANNs können für eine Vielzahl von Aufgaben verwendet werden. Um jedoch bei der für das ANN gewählten Aufgabe hohe Genauigkeitswerte erreichen zu können, gilt es das Modell bestmöglich für diese abzustimmen. Gleichzeitig soll ein Overfitting verhindert werden, bei welchem das Modell zu sehr an die Trainingsdaten angepasst ist und somit an Genauigkeit bei, für das Modell, fremden Datensätzen einbüßt. [103] Hierfür stehen unter anderem die sogenannten *"Hyperparameter"* zur Verfügung. Diese Hyperparameter bilden die im Trainingsprozess verwendeten Kennzahlen, welche in den folgenden Unterpunkten erklärt werden. [104]

4.4.1 Parameter

Um die Komplexität eines ANNs numerisch anhand einer Kennziffer darstellen zu können, werden die trainierbaren Parameter eines Modells als Modellgröße angegeben. Die Anzahl der Parameter ist von der Modellarchitektur abhängig und somit vor dem Trainingsprozess zu definieren. Trainierbare Parameter sind die Eingangswichtungen W_k und die Biases b_k eines Neurons k. Da ein neuronales Netz nicht vollständig verbunden sein muss, kann die Parameterzahl nicht ohne das Wissen der Verbindungsanzahl berechnet werden. [85, S. 51] Die Parameterzahl wird demnach von der Anzahl der verarbeitenden Schichten eines ANNs und der künstlichen Neuronen innerhalb einer verarbeitenden Schicht beeinflusst.

$$n_{Parameter} = n_{W_k} + n_{b_k}$$

Die Parameteranzahl ergibt sich aus der Anzahl der Eingangswichtungen und Biases.

$$(4 * 6) + (6 * 2) = 36$$

Anzahl der Parameter eines vollständig verbundenen FNNs ohne Biases
mit 4 Inputneuronen, einem Hidden-Layer mit 6 Neuronen und 2 Outputneuronen

Die Größe des Modells ist eine wichtige Kennzahl für ein ANN. Je nach Komplexität der Aufgabe ist eine andere Anzahl von Parametern zu trainieren. Kleine und große Modelle verfügen über Vor- und Nachteile. Kleine Modelle sind während der Backpropagation und Ausführung gering im Speicher- und Rechenzeitbedarf. Somit ist der Ressourcenbedarf direkt von der Modellgröße abhängig. Hierdurch werden kleine Modelle mit ein oder zwei Hidden-Layer für Schriftzeichenerkennung oder andere kontextunabhängige Aufgaben verwendet.

Große Modelle können auch komplexere Zusammenhänge verarbeiten. Für NLP-Aufgaben werden Modelle mit Parameteranzahlen im mindestens zweistelligen Millionenbereich verwendet. Mit einer höheren Anzahl zu trainierender Parameter steigen auch die benötigten Trainingsdaten und -schritte, wodurch auch die Trainingszeit und der Ressourcenbedarf steigen.

Doch ist das Wachstum der Genauigkeitskennziffern durch immer höherparametrigere Modelle zu erreichen? Das Problem der Überparametrisierung kann sich nachteilig auf die Leistungs- und Genauigkeitsmetriken auswirken. [103] Verfügt ein Modell über eine, für die zu erfüllende Aufgabe, zu hohe Größe, kann der Effekt des *“Assoziativen Speichers”* auftreten. Dieser Effekt beschreibt die Fähigkeit eines ANNs sich die Trainingsdaten lediglich einzuprägen, statt deren Wesensmerkmale zu erkennen. [105]

4.4.2 Batch-Size, Iterationen und Epoch

Moderne Rechentechnik erlaubt die parallele Ausführung vieler Prozesse. Dies kann durch die Verwendung von *“Batches”* genutzt werden, um die Verabeitungsgeschwindigkeit um ein vielfaches zu beschleunigen. So kann ein ANN die Wichtungen und Biases nach jedem verarbeiteten Batch anpassen. Je größer die Batch-Size gewählt wird, desto schneller ist der Trainingsprozess. Sind die Batches kleiner, wird das ANN häufiger aktualisiert. [106]

Während jede Ganzzahlige Batch-Size verwendet werden kann, empfiehlt es sich, bei 32 Daten pro Batch zu beginnen. Die mögliche Batch-Size ist vom Speicher der verarbeitenden Hardware abhängig. Je nach verfügbarer Hardware können Trainingsversuche mit einer Batch-Size von 64, 128, 256 oder noch höher gestartet werden. Befinden

sich für die zur Verfügung stehende Hardware zu viele Daten in einem Batch, wird ein *"Out-Of-Memory"*-Fehler (kurz: OOM) ausgegeben. [106]

Jeder Verarbeitungsschritt eines Batches wird *"Iteration"* genannt. Ein Epoch wird erreicht, wenn jeder Trainingsdatensatz ein Mal verarbeitet wurde. Demnach entspricht ein Epoch gleich der Batch-Size multipliziert mit der Iterationsanzahl. [106]

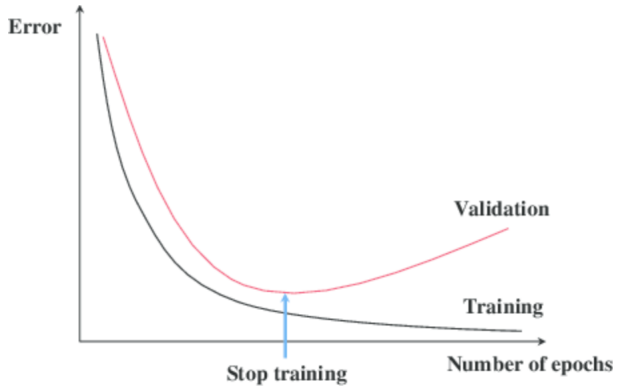

Abbildung 4.10: Early Stopping zur Verhinderung von Overfitting [107]

In Abbildung 4.10 wird ein Overfitting durch zu langes Training mittels der selben Datensätze dargestellt. Um diesen Effekt zu verhindern, wird mithilfe von *"Early Stopping"* das Training unterbrochen, sobald die Genauigkeitswerte für den Validierungsdatensatz sinken. [107]

4.4.3 Weight Decay

Im Kapitel 4.3 wurde die Learning Rate beschrieben. In der Backpropagation werden nach jedem Batch Wichtungen und Biases angepasst, aber was passiert mit den lange nicht angepassten Wichtungen? Diese könnten für die Klassifizierung unwichtig oder sogar hinderlich erscheinen. [108]

Um während des Trainingsprozesses die lange nicht angepassten Wichtungen zu modifizieren und die Komplexität eines ANNs zu reduzieren, wird ein *"Weight Decay"* eingeführt. Dieses senkt diese Wichtungen exponentiell in Richtung Null. [108]

4.5 Zusammenfassung

Für NLP-Aufgaben ist DL dem ML in vielen Aspekten vorzuziehen, wenn die Zeitkritikalität der Ausführung dies hergibt. Die Toxizitätserkennung protifiert von dem Genauigkeitsgewinn komplexer DL-Modellen. Die LSTM-Architektur konnte fast zwei Jahrzehnte lang SOTA-Modelle ermöglichen. Seit der Einführung der Transformer-Architektur im Jahr 2017 wurde die Mehrzahl der SOTA-Modelle auf dessen Basis erstellt. Transformer-Modelle bieten gegenüber LSTM-Modellen nicht nur eine höhere Genauigkeit, sondern auch eine deutlich schnellere Ausführung durch den Verzicht rückführender Schleifen.

Die Genauigkeit von DL-Modellen hängt von den Trainingsmethoden ab. Moderne Sprachmodelle werden in einer Mischung aus Un- und Supervised-Learning trainiert. Das Unsupervised-Learning wird ohne Labels, dafür aber mit Datensätzen, mindestens im Gigabytebereich, durchgeführt. Hierdurch soll dem Modell die Grundstruktur der zu lernenden Sprache angeeignet werden. Das Supervised-Learning ist für das Feintuning des Modells zuständig. Hierfür werden annotierte Daten benötigt. Für das Feintuning sind Daten im Megabytebereich für gute Genauigkeitswerte bereits ausreichend. Im Feintuning ist eine Anpassung der Hyperparameter notwendig, um ein Overfitting zu verhindern.

5 Anpassungen der Architekturen

Nachdem die Rahmenbedingungen für die modernen NLP-Modelle geklärt sind, gilt es die Anpassungen der Architekturen zu erläutern. Die Transformer-Architektur wurde von den Plattformbetreibenden für den Praxiseinsatz zur Erkennung von Hassrede und anderen Communityrichtlinien angepasst. Im Rahmen dieser Masterthesis werden jene Modelle betrachtet, welche in einer weiterführenden Arbeit über ein deutsches Pre-Training verfügen. Auch multilingual trainierte Modelle werden auf ihre Einsatzfähigkeit in der Toxizitätsbestimmung evaluiert.

Daher werden BERT, RoBERTa, XLM-R und GPT genau betrachtet. Während der Recherche für diese Masterthesis wurden ELMo, Grover, MegatronLM und XLNet für die Verwendung als Toxizitätsbestimmungsmodell im deutschen Sprachraum ausgeschlossen. Gründe hiervon sind entweder ein deutlich schlechteres Abschneiden des Modells in der Evaluierungsphase dieser Arbeit oder ein Fehlen eines deutschsprachigen Modells.

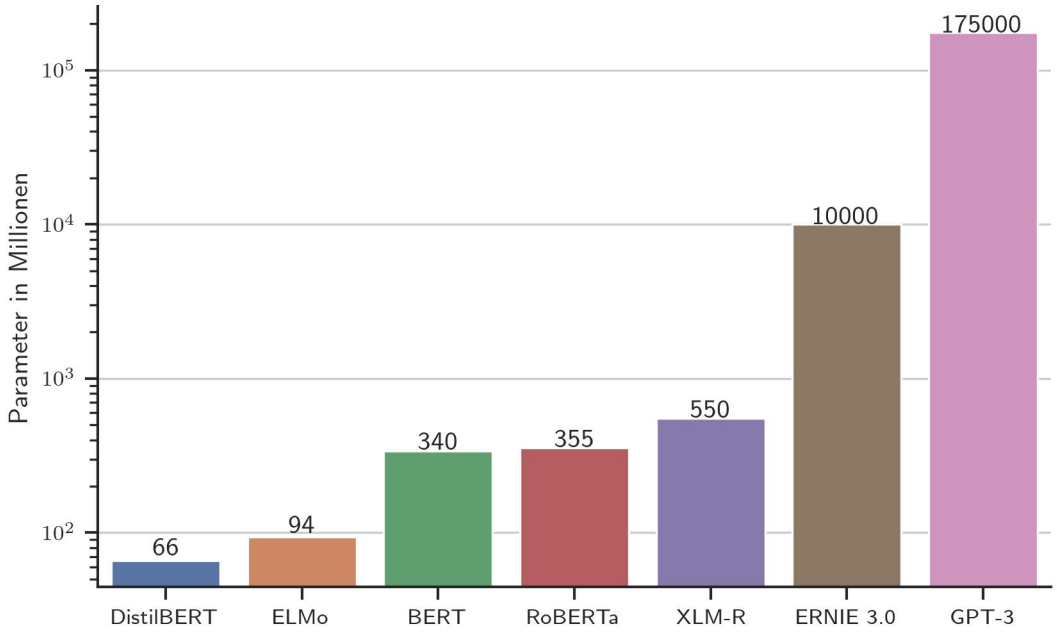

Abbildung 5.1: Parameteranzahlen von englischsprachigen Transformer-Modellen (Abbildung nach [109])

5.1 Google AI - BERT

Der *"Bidirectional Encoder Representation from Transformers"* (kurz: BERT) ist ein im Jahr 2018 von Google AI veröffentlichtes DL-Modell. [110, S. 1] BERT ist in 25 verschiedenen Größen verfügbar, mit einer Reichweite von zwei bis zwölf Encodern und 128 bis 768 Hidden-Layer. Die beiden größten verfügbaren Modelle sind `bert-base` und `bert-large`. `bert-base` ist 110 Millionen Parameter groß und verfügt über 12 Encoder mit insgesamt 768 Hidden-Layer. `bert-large` ist 340 Millionen Parameter groß und verfügt über 24 Encoder mit insgesamt 1024 Hidden-Layer. [110, S. 3]

Um die Konzepte von Sprache und Kontext in BERT darstellen zu können, wird vor der Anpassung des Modells an die Aufgabe ein für die Sprache allgemeingültiges Pre-Training durchgeführt. Anschließend kann mit dem Feintuning das vortrainierte BERT-Modell an die spezifische Aufgabe angepasst werden. [110, S. 1] Da ein Pre-Training ressourcen- und zeitintensiv ist, sind auf verschiedenen Plattformen vortrainierte Meilensteine der Modelle verfügbar.

Das Pre-Training von BERT verwendet zwei Datensätze, die zusammen etwa 16 GB unkomprimierte, englischsprachige Wörter beinhalten. Der *"BooksCorpus"* macht hiervon 800 Millionen Wörter aus. Der von Google AI verwendete englischsprachige Wikipedia-Dump liefert dem Modell die restlichen 2,5 Milliarden Wörter und verzichtet hierbei auf Daten aus Listen, Tabellen und Überschriften. [110, S. 5]

Was implementiert BERT in das Pre-Training, was andere Modelle noch nicht nutzten? Um die Genauigkeit zu erhöhen, verwendet Google AI während des Pre-Trainings zwei von dem ANN zu lösende Aufgaben. Mit dem Masking soll das Modell einen Token innerhalb eines Satzes ausfüllen. Während des Pre-Trainings werden 15% aller Token mit einem `[MASK]`-Token versehen.

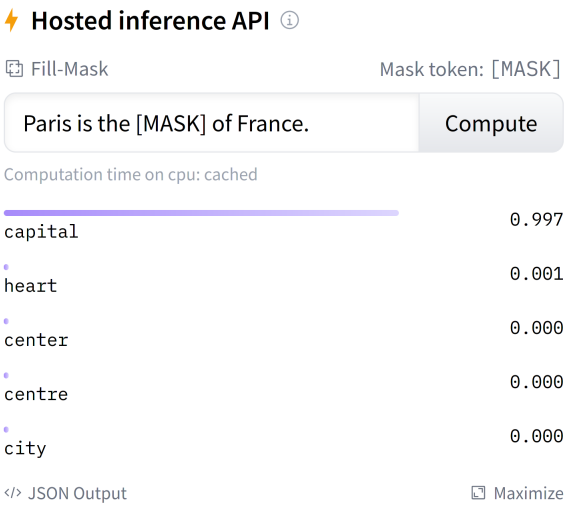

Abbildung 5.2: Beispiel einer Maskierungsausgabe mit `bert-base-uncased` während dessen Pre-Trainings [111]

Die zweite Aufgabe, während des Pre-Trainings von BERT, ist die *"Next Sentence Prediction"* (kurz: NSP). Hierbei werden dem Modell zwei Sätze übergeben, von denen es die richtige Reihenfolge bestimmen soll. Die möglichen Label dieser Aufgabe sind `IsNext` oder `NotNext`. Die Sequenzlänge darf bei BERT nicht 512 Token überschreiten. Hierbei ist die Anzahl der Sätze und Wörter pro Sequenz unerheblich. Dies wird *"Segment-Pair+NSP"* genannt. Lange und zusammengesetzte Wörter können, je nach Konfiguration, in mehrere Token aufgeteilt werden. Zusammenhängende Token werden per `##` gekennzeichnet. Wichtig hierbei ist, dass die Konfiguration der Tokenisierung in der Anwendung mit dem Pre-Training und Feintuning übereinstimmt. [110, S. 4–5]

$$\begin{aligned}\text{Input} &= \texttt{[CLS] the man went to [MASK] store [SEP]}\\ &\quad\ \texttt{he bought a gallon [MASK] milk [SEP]}\\ \text{Label} &= \texttt{IsNext}\end{aligned}$$

$$\begin{aligned}\text{Input} &= \texttt{[CLS] the man [MASK] to the store [SEP]}\\ &\quad\ \texttt{penguin [MASK] are flight ##less birds [SEP]}\\ \text{Label} &= \texttt{NotNext}\end{aligned}$$

Abbildung 5.3: Beispiel einer NSP-Aufgabe während des Pre-Trainings eines BERT-Modells [110, S. 2]

Nach dem Pre-Training verfügt das Modell über ein allgemeines Sprachverständnis. Doch wie kann dieses verwendet werden, um eine Aufgabenstellung zu lösen? Mithilfe des Feintunings kann durch Supervised-Learning das Modell angepasst werden. Dieser Prozess ist vielfach ressourceneffizienter als das Pre-Training, da nur geringe Änderungen an dem ANN vorgenommen werden. So werden durch das Feintuning mindestens die Ausgangsparameter angepasst und höchstens kleine Änderungen innerhalb der Encoder vorgenommen. [110, S. 5]

BERT ist auch als vortrainierter Meilenstein in anderen Sprachen verfügbar. Beispiele hierfür sind `german BERT`, `AraBERT`, (arabisch) und `mBERT` (104-sprachig). Außerdem veröffentlichte die Bayerische Staatsbibliothek - Münchener Digitalisierungszentrum (kurz: dbmdz) diverse deutsche BERT-Meilensteine. Weitere Forschungsarbeiten von Plattformbetreibenden werden in den folgenden Unterpunkten dieser Arbeit erläutert. Interessante Weiterentwicklungen von BERT sind unter anderem GAN-BERT und DistilBERT.

DistilBERT

Das Ziel der Entwicklung von DistilBERT ist nicht die Genauigkeit von BERT zu verbessern oder die benötigten Trainingsdaten für das Feintuning zu reduzieren, sondern die Modellgröße zu verringern und somit die Ausführungs-, Trainingsgeschwindigkeit und den Ressourcenbedarf zu reduzieren. Hierbei soll der Genauigkeitsverlust so gering wie möglich ausfallen. [112] Um dieses Ziel zu erreichen Als Ergebnis ist die Ausführungszeit von

`distilbert-base-uncased` gegenüber `bert-base` 60% schneller und die Modellgröße 40% kleiner. Diese Verringerung wurde mittels einer Komprimierung von BERT erreicht. Hierbei wurden alle Neuronen entfernt, deren Wichtungen nahe Null sind. [113, S. 1–2] Trotz dieser Simplifizierung des BERT-Modells erreicht DistilBERT lediglich durchschnittlich 3% geringere Genauigkeitsmetriken als BERT. [113, S. 3]

GAN-BERT

Bei GAN-BERT handelt es sich um ein BERT-Modell, welches ein Feintuning per GAN und somit per Semi-Supervised-Learning statt Supervised-Learning durchführt. Diese Vorgehensweise soll besonders bei Aufgaben helfen, die über einen binären Klassifikator hinaus gehen. BERT kann die Genauigkeitswerte von GAN-BERT erreichen, jedoch werden hierfür durchschnittlich doppelt so viele annotierte Trainingsdatensätze im Feintuning benötigt. Die Genauigkeitsvorteile von GAN-BERT gegenüber BERT sind in Abbildung 5.4 dargestellt. [114, S. 2114–2117]

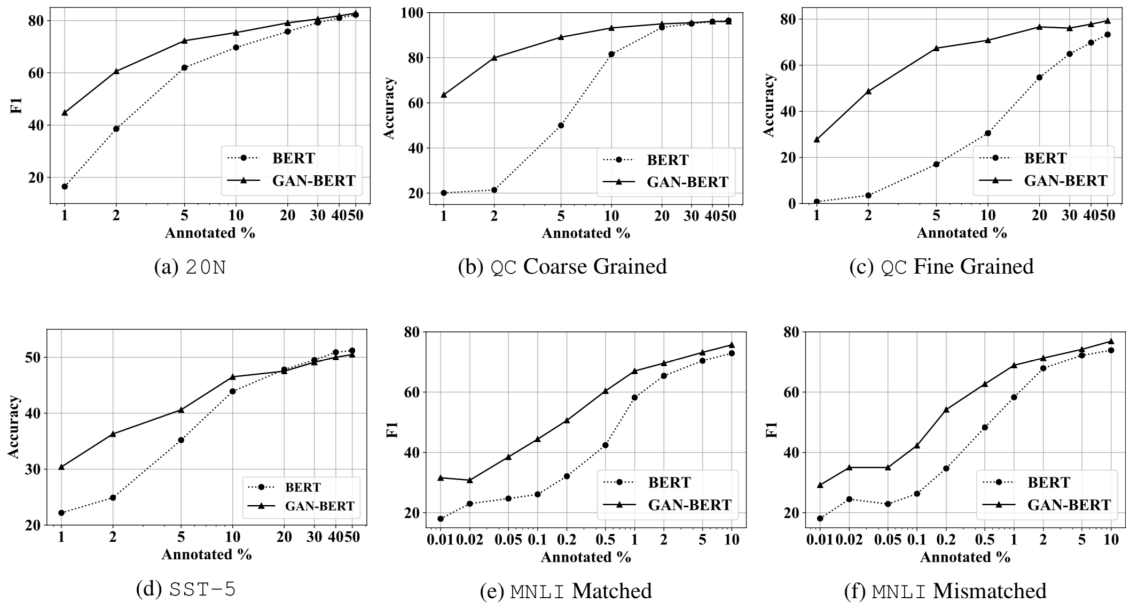

Abbildung 5.4: Genauigkeitsvorteile von GAN-BERT gegenüber BERT bei Semi-Supervised-Learning über verschiedene Anteile annotierter Daten [114, S. 2117]

Doch wie funktioniert ein GAN in NLP-Aufgaben? In Abbildung 5.5 ist der Aufbau des Trainingsprozesses mittels GAN-BERT dargestellt. Der Generator erhält Zufallszahlen und gibt aus diesen sprachrepräsentierende Vektoren aus. Ziel des Generators ist, die sprachrepräsentierenden Vektoren möglichst ununterscheidbar zu menschlichen Eingangsdaten zu gestalten. BERT erhält für das Feintuning sowohl annotierte, als auch nicht-annotierte Daten. Die Aufgabe des Discriminators ist die Zuordnung der Eingangsdaten zu den einzelnen Klassen. Des Weiteren soll dieser zwischen menschlichen und zufallsgenerierten Eingangsdaten unterscheiden. [114, S. 2115–2116]

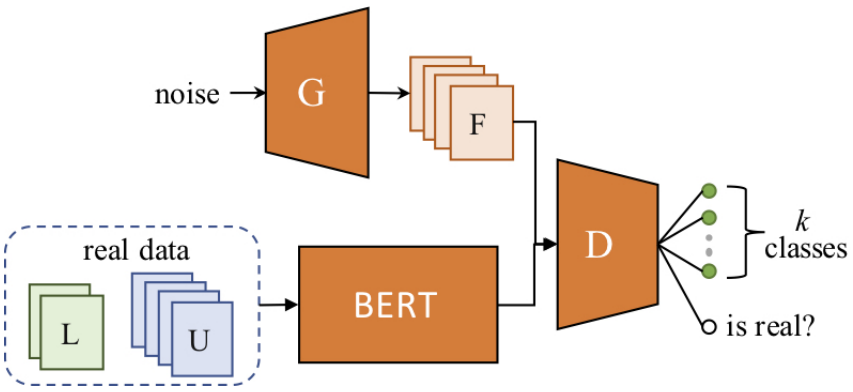

Abbildung 5.5: Erweiterung der BERT-Architektur um ein GAN, bestehend aus Generator G und Discriminator D [114, S. 2116]

Für die weiteren Analysen von BERT-Modellen im Rahmen dieser Masterthesis werden german BERT und die BERT-Modelle von der dbmdz verwendet. Des Weiteren wird die Einsatzfähigkeit von GAN-BERT bei deutschsprachig vortrainierten Modellen geprüft.

5.2 Facebook Research - RoBERTa

"Robustly optimized BERT pretraining approach" (kurz: RoBERTa) ist eine erweiterte Trainingsmethode für BERT. Die Architektur entspricht der von BERT, welcher lediglich, je nach Modellgröße, 15 - 20 Millionen Neuronen hinzugefügt wurden. Diese zusätzlichen Neuronen werden für ein fast doppelt so großes Byte-Level-Byte-Pair-Encoding-Vokabular verwendet, welche in der Untersuchung keinen Genauigkeitsgewinn gegenüber BERT mit sich bringt. [115, S. 6] Die Trainingsdaten sind deutlich vielseitiger als es bei BERT der Fall ist. So beinhaltet RoBERTa zusätzlich CC-News, was der englischsprachige Teildatensatz des CommonCrawl News Datensatzes ist. Die Größe hiervon beträgt 76GB. Weiterhin wird der OpenWebText-Datensatz mit einer Größe von 38GB und der Stories-Datensatz, welcher ebenfalls ein Teil des CommonCrawl-Datensatzes ist, mit einer Größe von 31 GB. Somit sind die Pre-Trainingsdaten von RoBERTa gegenüber BERT mit 160GB gegen 16GB 10-fach größer. [115, S. 3]

Die Evaluierung des Modells wurde per *"GLUE"*, *"Stanford Question Answering Dataset"* (kurz: SQuAD) und *"ReAding Comprehension from Examinations"* (kurz: RACE) durchgeführt. Der Pre-Trainingsprozess ist in RoBERTa dahingehend angepasst, dass dynamische statt statischer Masken verwendet werden. Dynamische Masken bedeuten, dass bei jedem Trainingsvorgang ad-hoc Masken für die Sätze generiert werden, anstatt dass diese bei dem Beginn des Trainingsprozesses bereits feststehen. Die dynamischen Masken erzielen in der Evaluierung eine Modelloptimierung von durchschnittlich $0,1\overline{6}$ Prozentpunkten. [115, S. 4] Der Genauigkeitsgewinn in den einzelnen Benchmarks ist in der Tabelle 5.1 dargestellt.

Masking	SQuAD 2.0	MNLI-m	SST-2
BERT	76,3	84,3	92,8
statisch	78,3	84,3	92,5
dynamisch	78,7	84	92,9

Tabelle 5.1: F1-Genauigkeiten von RoBERTa bei statischer und dynamischer Maskierung mit Referenz zu BERT. (Tabelle nach [115, S. 4])

Für die Vorverarbeitung der Eingangsparameter wurden neben der in BERT verwendeten Segment-Pair+NSP weitere Methoden evaluiert. Bei keiner Vorverarbeitungsmethode überschreitet die Anzahl der maximalen Token pro Eingangsparameter 512. Bei Sentence-Pair+NSP werden zwei aufeinanderfolgende Sätze aus dem gleichen oder zwei verschiedenen Dokumenten entnommen. Bei Full-Sentences werden so lange vollständige Sätze aufeinanderfolgende Sätze aus beliebig vielen Dokumenten entnommen, bis die maximal erlaubte Anzahl von 512 Token erreicht ist. Bei Doc-Sentences werden die vollständigen aufeinanderfolgenden Sätze als Eingangsparameter übergeben, die innerhalb eines Dokumentes auftreten. [115, S. 5]

Die höchste Genauigkeit wurde mit der Doc-Sentences-Methode erlangt. Der Genauigkeitsgewinn gegenüber BERT beträgt 1,625 Prozentpunkte. Aus architektonischen Gründen wurde Full-Sentences mit einem Genauigkeitsgewinn von 1,25 Prozentpunkten gegenüber BERT für die Entwicklung von RoBERTa gewählt. [115, S. 5]

Die Batchgröße der Eingangsdaten kann ebenfalls einen Unterschied für das zu trainierende Modell ausmachen. Bei dem Training von RoBERTa wurde eine Batchgröße von 2048 als am effizientesten evaluiert. Dies wird durch die hohe Anzahl von GPUs während des Trainingsprozesses ermöglicht. [115, S. 5] Die Batchgröße in BERT beträgt, je nach Aufgabe, lediglich 16 oder 32. [110, S. 6]

Zusammenfassend können die, für die Genauigkeit gewinnbringenden, Änderungen von RoBERTa gegenüber BERT mit der dynamischen Maskierung, Full-Sentences und größeren Batches begründet werden. RoBERTa ist auch als vortrainierter Meilenstein in anderen Sprachen verfügbar. Beispiele hierfür sind `CamemBERT` (französisch), `UmBERTo` (italienisch) und `GottBERT` (deutsch).

Seit Dezember 2020 existiert das deutschsprachige RoBERTa-Modell GottBERT. Dieses wurde mit 145GB deutschsprachigen Texten trainiert. Das Pre-Training von GottBERT wurde mit 256 TPUs, 8192 großen Batches, mit 100.000 Schritten, einer Learning-Rate von 4e-4 und einem Weight-Decay durchgeführt. [116, S. 2–3] Für weitere Analysen von RoBERTa für die deutschsprachige Toxizitätsbestimmung innerhalb dieser Arbeit wird GottBERT verwendet.

5.3 Facebook Research - XLM

Basierend auf den Erkenntnissen des monolingual-trainierten RoBERTa-Modells entwickelte Facebook ein Verfahren für multilingual-trainierte Modelle. [117] Der Name XLM setzt sich aus Masked Language Model (kurz: MLM) und Cross-Lingual (kurz: XL) zusammen. Das Ziel von XLM ist es, die multilinguale Wissensdarstellung zu verbessern. Besonders bei Sprachen, für die wenige Trainingsdatensätze verfügbar sind, schneidet XLM besser als mBERT ab. [118, S. 4–5]

Das XLM-Framework unterstützt sowohl monolinguale, als aus XL-Modelle. Ein deutsches vortrainiertes Modell steht nicht zur Verfügung, weshalb XL-Modelle betrachtet werden. Das für 100 Sprachen trainierte XLM-R-Modell basiert auf RoBERTa und verfügt über 550 Millionen Parameter. [119] XLM-R wurde mit 500 NVidias V100 GPUs trainiert. Die Modellgröße ist enorm, weshalb es in vielen Szenarien mit nicht für den Livebetrieb ausreichenden Geschwindigkeiten ausgeführt werden kann. [120]

Das Limit multilingualer Modelle ist laut Conneau et al. 2019 das mögliche Vokabular. Dies beträgt bei XLM-R 250.000 Token. [118] Ein größeres Vokabular ist jedoch lediglich mit mehr Hardwareressourcen, wie schneller Speicheranbindung an die GPU/TPU oder unter diesen Recheneinheiten möglich. Sind die Kommunikationswege unter den Recheneinheiten schneller, können auch mehr dieser in einem Verbund genutzt werden, ohne auf einen Flaschenhalseffekt zu treffen. [120]

5.4 OpenAI - GPT

Generative Pre-Training (kurz: GPT) ist eine Pre-Trainingsmethode, welche ein vielseitig einsetzbares Modell trainieren soll. Hierbei soll das Feintuning für die einzelnen Aufgaben geringer als bei anderen Modellen ausfallen. Die GPT-Modelle sind bereits ohne Feintuning in der Lage viele verschiedene Aufgabenstellungen mit einer höheren Präzision als vorherige SOTA-ML-Modelle zu lösen. Bereits das erste GPT-Modell konnte ohne Feintuning in neun von zwölf evaluierten NLP-Aufgaben das neue SOTA-Modell werden. [121]

GPT-1 wurde im Juni 2018 vorgestellt und ist somit das erste große Modell auf Basis der Transformer-Architektur. [122] Der über 7.000 Bücher große BooksCorpus dient als Trainingsdatensatz des ersten GPT-Modells. [122, S. 4] Mit 117 Millionen trainierbaren Parametern entspricht es etwa der Größe von BERT. [121]

Im Februar des Folgejahres 2019 wurde GPT-2 vorgestellt und erschien somit noch sieben Monate vor BERT. [123] Als Datensatz wurde statt des BooksCorpus auf Webseiten zurückgegriffen. Um die Datenqualität zu wahren, wurde auf CommonCrawl verzichtet und stattdessen Reddit auf Posts mit mindestens 3 Karma gecrawlt, welche auf externe

Webseiten verlinken. Der hierdurch entstandene *“WebText”*-Corpus beinhaltet 8 Millionen einzigartige Webseiten mit insgesamt 40 GB Text. Das Team entfernte alle Wikipedia-Inhalte, damit in der Evaluierungsphase keine Trainingsdaten für Tests verwendet werden. [123, S. 3] GPT-2 verfügt über 1,5 Milliarden trainierbare Parameter und ist somit zehnfach größer als das Vorgängermodell GPT-1 und über vier Mal so groß wie RoBERTa. In Unsupervised-Learning-Aufgaben schneidet GPT-2 in sieben von acht Benchmarks besser als vorherige SOTA-Modelle ab. [121]

Eine interessante Beobachtung während der Evaluierung von GPT-2 ist die Tatsache, dass die bereits riesige Größe des Modells gegenüber anderen Modellen noch nicht das Limit für bessere Genauigkeitsmetriken ist. [123, S. 9] Mit diesem Wissen stattet OpenAI GPT-3 im Jahr 2020 mit 175 Milliarden Parametern aus und behält viele Eigenschaften von GPT-2 bei. Somit zählt es zu den größten DL-Modellen überhaupt. [121] Die Trainingsdaten wurden aus CommonCrawl, WebText2, Books1, Books2 und Wikipedia entnommen. Je nach der Qualität der Einträge, wurden die Daten während des Trainings mit unterschiedlichen Wichtungen versehen. GPT-3 stellt nicht in jeder Aufgabe SOTA-Ergebnisse auf, verbessert jedoch in fast jeder Disziplin die Zero- oder Few-Shot-Performance, bei welchen kein oder wenig Feintuning zum Einsatz kommt. [124]

Von GPT-2 existiert ein deutscher Pre-Training-Meilenstein von der dbmdz. Dieser wird für weitere Evaluierungen der GPT-Modelle im Rahmen dieser Masterthesis verwendet.

Teil II

Zusammenführen der Erkenntnisse und Modellerarbeitung

6 Auswahl geeigneter Klassifikatoren

Da alle vorgestellten Klassifikatoren mit Python angesprochen werden können und die dort verwendeten Werkzeuge über gute Dokumentationen verfügen, wird Python als einheitliches Werkzeug für alle Verarbeitungsschritte verwendet. [125, 126, 127]

6.1 Sprachverständnis

Im Rahmen dieser Masterthesis werden Modelle mit Verständnis für wenige Sprachen bevorzugt. Je mehr Sprachen ein Modell unterstützt, desto größer muss es für eine ausreichende Genauigkeit in allen unterstützten Sprachen ausfallen. Monolinguale Modelle können meist mit ausreichender Geschwindigkeit mit einer GPU oder TPU ausgeführt werden. Bei kleinen Modellen reicht sogar eine CPU für die Verarbeitung einer Anfrage unter einer Sekunde aus. Facebooks XLM-R-Modell unterstützt 100 Sprachen, was den Rechenbedarf bei der Anwendung des Modells vervielfacht. Um den Rechenaufwand zu begrenzen, wird XLM-R in dieser Masterthesis nicht evaluiert.

Das Anlernen des Sprachverständnisses erfolgt in den meisten Modellen in einer Mischung aus Supervised- und Unsupervised-Learning. Mit dem Modell *“Contrastive BERT for Reinforcement Learning”* (kurz: CoBERL) wird die Anpassungsfähigkeit eines Hybriden aus LSTM- und Transformer-Architektur an das Reinforcement Learning bewiesen. [128] Da keines in der Recherche gefundenen Modelle Reinforcement Learning für das Feintuning in Sprachklassifizierungsaufgaben verwendet, wird auf ein diesen Ansatz im weiteren Verlauf dieser Masterthesis verzichtet.

Das generelle Sprachverständnis eines Modells erfolgt zum größten Anteil mithilfe von Unsupervised-Learning. Aus der Recherche englischsprachiger Modelle kann geschlussfolgert werden, dass Modelle mit größerem Pre-Trainingsdatensatz höhere Genauigkeitswerte bei Klassifizierungsaufgaben nach dem Feintuning erreichen. In der englischen Sprache ist die Größe der hierfür geeigneten Trainingsdaten deutlich höher als in der deutschen Sprache. Da bereits fertige multilinguale Modelle der Plattformbetreibenden in ihrer bisherigen Natur und der aktuell zur Verfügung stehenden Rechentechnik nicht wirtschaftlich anwendbar sind, kommen diese nicht als Ersatz eines komplett in der deutschen Sprache vortrainierten Modells infrage.

Unter den Vortrainierten deutschsprachigen Modellen zählt das auf RoBERTa basierende GottBERT mit etwa 145 GB Trainingsdaten zu dem quantitativ besten vortrainierten deutschsprachigen Transformer-Sprachmodell. Da RoBERTa mit einem Score von 84,6 in dem SuperGLUE-Benchmark wesentlich bessere Ergebnisse als BERT mit 69,0 erzielt,

besteht für ein deutschsprachiges Modell ebenfalls das Potential, Genauigkeitswerte über die von deutschsprachigen BERT-Modellen zu erreichen. [129]

6.2 Gegenüberstellung der Klassifikatoren

In der Tabelle 6.1 befinden sich die deutschsprachig vortrainierten Meilensteine, welche frei zum Download zur Verfügung stehen. Diese Modelle unterscheiden sich besonders in den folgenden Kriterien.

Die Orthografie beschreibt die Fähigkeit, zwischen Groß- und Kleinschreibung unterscheiden zu können. Hierbei steht `cased` für die Möglichkeit großer Buchstaben in den Token und `uncased` lediglich für kleine Buchstaben in den Token. Außerdem werden Umlaute lediglich in `cased`-Modellen standardmäßig korrekt dargestellt. Manche Tokenisierer verfügen über einen Parameter namens `strip_accents`, bei welchem mit Wert `False` Umlaute auch in `uncased`-Modellen dargestellt werden können.

Die Modellarchitektur, und somit die Anzahl der Parameter, Encoder und Hidden-Layer innerhalb der Encoder wird von dem zugrunde liegenden englischsprachigen Modell vorgegeben. Die deutschsprachig vortrainierten Modelle entsprechen der Architektur der Basismodelle von BERT, DistilBERT, RoBERTa und GPT-2. Rechenaufwändigere deutschsprachige Modelle auf Basis von `bert-large` und `roberta-large` konnten in der Recherche dieser Masterthesis nicht ausfindig gemacht werden.

Die Vokabulargröße bestimmt ebenfalls die mögliche Genauigkeit eines Modells. Ein größeres Vokabular kann bei längerem Training zu besseren Verarbeitungsergebnissen führen. Besonders Modelle mit Unterstützung für Orthografie können von einem größeren Vokabular profitieren. Im Umkehrschluss ist es wahrscheinlich, dass Modelle bei kleinem Vokabular eine höhere Genauigkeit ohne Orthografie erzielen.

Modell	bert-base-german	distilbert-base-german	bert-base-german-dbmdz	german-gpt2	gottbert-base
Orthografie	cased	cased	un-/cased	cased	cased
Parameter	110M	66M	110M	125M	126M
Pre-Trainingsdaten	12GB	51GB	16GB	16GB	145GB
Hidden-Layer	768	768	768	768	768
Encoder	12	6	12	12	12
Vokabulargröße	30k	31k	31k	50k	52k

Tabelle 6.1: Vergleich deutschsprachig vortrainierter Modellmeilensteine

6.3 Zusammenfassung

In der deutschen Sprache stehen mehrere Modelle zur Verfügung, die allesamt nicht die großen Modellgrößen verwenden. In Python stehen mit PyTorch und der Transformers-Bibliothek von Huggingface geeignete Werkzeuge für das Feintuning von Sprachmodellen zur Verfügung. Ein Feintuning kann sowohl per Supervised-Learning, als auch per Semi-Supervised-Learning via GAN stattfinden. Bei welchen Modellen dies möglich ist und wie hoch der Genauigkeitsunterschied hierdurch ausfällt, wird in der Modelloptimierung evaluiert.

7 Modelloptimierung

7.1 Trainingsdaten

Für das Feintuning der Modelle werden die in der Einleitung beschriebenen Daten von GermEval-Challenges verwendet. Gründe hierfür sind die Nähe an der Toxizitätsdefinition dieser Masterthesis und die hohe Quali- und Quantität der GermEval-Daten.

Für ein Modell ist es jedoch nicht möglich, jedes als toxisch annotiertes Datum als solches zu erkennen. Einige als toxisch annotierte Daten benötigen weitere Kontextmerkmale, um diese als toxisch erkennen zu können. Ein Beispiel hierfür ist der Kommentar *“Kann sich nur noch um gramm [sic!] handeln”*, welcher als toxisch annotiert wurde. Ohne Kontextmerkmale kann das Modell nicht einordnen, was sich um Gramm handeln soll. Somit keine genaue Analyse dieses Inhalts möglich ist.

Für Kommentare wie *“Ich will nen neuen Deutschen Kaiser”* fehlt KI-Modellen zudem die historische Einordnung von Inhalten und die Übertragung dieser in aktuelle Debatten. Auch ist die Abgrenzung zwischen toxisch und nichttoxisch in den annotierten Daten nicht eindeutig. Auch sind Kommentare wie *“Die Deppen der Nation....Österreicher lachen am lautesten....”* oder *“Wetten.du bist so ein verbittertes scheidungskind [sic!]”* beispielsweise als nicht toxisch annotiert.

Diese Faktoren bewirken, dass das Lösen dieser Klassifizierungsaufgabe durch Menschen keine Genauigkeit von 100% erreichen kann. Daher wird, wie beispielsweise beim SuperGLUE-Benchmark, eine menschliche Genauigkeit von 90% angenommen. [129]

7.2 Auswahl eines Klassifikators

Für die Evaluierung der verfügbaren Modelle für die Toxizitätserkennung im deutschen Sprachraum, bietet sich die Nutzung von Google Colab an. Dieses ist eine Remote-Python-Umgebung im Webbrowser, bei welcher eine GPU oder TPU eingebunden werden kann. In der kostenlosen Version stehen lediglich schwächere GPUs und keine TPUs zur Verfügung. Im Rahmen dieser Masterthesis wird Colab Pro für eine größere Flexibilität im Feintuning der Modelle verwendet.

Nach dem Import der PyTorch-, TensorFlow-, und Transformers-Bibliotheken ist die Umgebung bereit für das Training eines ANNs. Zur Unterstützung in der Datenaufbereitung und der Auswertung der Genauigkeitsmetriken werden unter anderem *“numpy”*, *“sklearn”* und *“pandas”* verwendet.

Im Anhang A befindet sich der Supervised-Learning-Code, mit welchem die Modelle auf die Klassifizierungsaufgabe angepasst werden. Die Tabelle 7.1 bietet eine Übersicht der in der Einleitung genannten Genauigkeitsmetriken des jeweiligen Modells. Einige Modelle des dbmdz befinden sich mehrfach in der Transformers-Bibliothek von Huggingface. Doppelte und ältere Versionen der gewählten Modelle befinden sich aufgrund ihrer zu starken Ähnlichkeit zu den gewählten Versionen nicht in der Vergleichstabelle.

Für das Feintuning aller ANNs werden die Hyperparameter verwendet, die bei den größten Anteil der Modelle die besten Ergebnisse liefern. So ergibt sich aus der Recherche heraus eine Empfehlung für 4 Epochs. In Versuchen während der Modellevaluierung bestätigt sich diese Zahl als Empfehlenswert. Werden weniger Epochs für das Training verwendet, sinkt die Qualität der Klassifizierung für den Trainings-, als auch den Testdatensatz. Bei mehr als 4 Epochs tritt bereits eine übermäßige Anpassung des Modells an die Trainingsdaten auf und es kommt zum Overfitting des Modells auf den Trainingsdatensatz. Das Overfitting ist an einer sinkenden Genauigkeit für den Testdatensatz erkennbar, während die Genauigkeit für den Trainingsdatensatz weiter steigt. Hierdurch bestätigt sich die positive Auswirkung des Early Stoppings in dieser Klassifizierungsaufgabe bereits nach 4 Epochs.

Die Batchgröße für diesen Versuch beträgt 32. Dieser Wert ist ein guter Mittelwert zwischen Ausführungsgeschwindigkeit und Ressourcenbedarf. Mit einer Batchgröße von 32 ist das Feintuning auch möglich, wenn der Google-Colab-Sitzung eine weniger leistungsstarke GPU zugewiesen wird.

Um den Ressourcenbedarf weiter zu reduzieren, beträgt die maximale Sequenzlänge für die Evaluierung des besten Feintunings 64. Dies bedeutet, dass die verarbeiteten Kommentare nach 64 Token abgeschnitten werden. Kürzere Kommentare werden mit Padding-Token auf eine Länge von 64 erweitert. Weiterhin wird die häufig verwendete Learning-Rate von 2e-5 auch für diesen Versuchsaufbau verwendet.

Die Tabelle 7.1 zeigt, dass die auf BERT-basierenden Modelle im deutschen Sprachraum die besten Ergebnisse in der Toxizitätsklassifizierung erzielen. Um eine Einordnung der Ergebnisse mit aktiv eingesetzten Werkzeugen zu ermöglichen, befinden sich neben den Genauigkeiten der herunterladbaren Modelle auch die Genauigkeitswerte der Perspective API von Google Jigsaw. Der Code für die Verarbeitung mit der Perspective API befindet sich in Anhang C.

Die geringsten Genauigkeitsmetriken weisen die deutschsprachig trainierten RoBERTa- und GPT-2-Modelle auf. Doch wie ist das, besonders bei der höheren Modell- und Trainingsdatengröße von GottBERT, zu erklären? Sowohl RoBERTa, als auch GottBERT wurden mit einer vielfachen Batchgröße trainiert, als mit der in Google Colab zur Verfügung stehenden GPU möglich ist, ohne einen OOM-Fehler zu erhalten. Weitere Faktoren können ein zu kurzes Pre-Training von GottBERT oder eine zu niedrige Datenqualität in

Modell	Accuracy	Precision	Recall	F1	MCC
Jigsaw Perspective API	78.10%	68.51%	**65.54%**	75.31%	50.65%
distilbert-base-german-cased	69.94%	57.31%	42.61%	63.74%	28.76%
dbmdz/distilbert-base-german-europeana-cased	69.45%	56.19%	46.17%	64.36%	29.31%
bert-base-german-cased	77.71%	70.95%	57.57%	73.73%	48.25%
bert-base-german-dbmdz-cased	78.93%	74.41%	57.39%	74.87%	50.95%
bert-base-german-dbmdz-uncased	**80.25%**	**76.79%**	59.57%	**76.48%**	**54.15%**
dbmdz/bert-base-german-europeana-cased	75.70%	66.03%	56.96%	71.64%	43.68%
dbmdz/bert-base-german-europeana-uncased	76.16%	75.07%	44.00%	69.58%	43.34%
uklfr/gottbert-base	68.39%	53.85%	43.83%	62.72%	26.07%
dbmdz/german-gpt2	69.14%	56.36%	39.30%	62.34%	26.45%

Tabelle 7.1: Genauigkeitsmetriken der GermEval 2018 Shared Task 1 mit 4 Epochs, Batchgröße 32, 2e-5 Learning-Rate und einer Sequenzlänge von 64. Keine Vorverarbeitung und kein Warmup. Perspective API als Referenzwert.

diesem sein. Durch die geringere Modellgröße schneiden auch die DistilBERT-Modelle schlechter ab als die BERT-Modelle.

Unter den BERT-Modellen erreichen die Modelle der dbmdz die höchsten Genauigkeitswerte. Die Version mit Orthografie erreicht einen MCC von 50,95%, während die Version ohne Orthografie mit einem MCC von 54,15% noch besser abschneidet. Die von dem gleichen Entwickler trainierten *"europeana"*-Modelle verfügen zwar über etwa vierfach so viele Pre-Trainingsdaten, schneiden jedoch mit einem MCC von etwa 43,5% schlechter in der Evaluierung ab als die Grundmodelle der dbmdz. Außerdem generiert das von deepset vortrainierte `bert-base-german-cased`-Modell mit einem MCC von 48,25% hohe Genauigkeitswerte. Für weitere Analysen kommen daher die Grundmodelle von der dbmdz und das Modell von deepset infrage.

Doch wie wirkt sich ein Training via GAN auf die Genauigkeitsmetriken der Modelle in der Klassifizierungsaufgabe toxischer Kommentare aus? Im Anhang B befindet sich der Code für das Semi-Supervised-Learning mit GAN-BERT. Dieser Versuchsaufbau verwendet die Hyperparameter aus dem GAN-BERT-Paper, um die Vorteile der GAN-Architektur zu nutzen. [114] Die Unterschiede des Trainings mittels Semi-Supervised-Learning via GAN bedeuten in den Hyperparametern, dass weniger Epochs und weniger annotierte Trainingsdaten verwendet werden. Die geringere Anzahl der Trainingsschritte erfordert eine höhere Learning-Rate für GAN-Epochs, welche für diesen Versuchsaufbau 5e-5 statt 2e-5 beträgt. Des Weiteren wird ein Warmup verwendet, welches 10% der annotierten Trainingsdatengröße entspricht. Das Warmup soll dabei helfen, den Startzustand des ANNs für das Feintuning besser auszuwählen.

Die Tabelle 7.2 bietet eine Übersicht der ausgewählten deutschsprachigen Modelle und die dazugehörigen Genauigkeitsmetriken für die GermEval 2018 Shared Task 1. GAN-BERT ist in seiner Implementierung nicht mit GottBERT, DistilBERT und GPT-

Modell	Accuracy	Precision	Recall	F1	MCC
bert-base-german-cased	74.71%	67.60%	48.39%	69.29%	40.36%
bert-base-german-dbmdz-cased	76.21%	64.48%	66.00%	73.57%	47.16%
bert-base-german-dbmdz-uncased	**77.68%**	65.98%	**70.18%**	**75.44%**	**50.96%**
dbmdz/bert-base-german-europeana-cased	71.31%	**76.85%**	21.71%	57.77%	29.56%
dbmdz/bert-base-german-europeana-uncased	72.29%	75.06%	27.03%	60.87%	32.46%

Tabelle 7.2: Genauigkeitsmetriken der GermEval 2018 Shared Task 1 mit einem GAN-Epoch, Batchgröße 32, 5e-5 Learning-Rate, 10% Warmup der annotierten Daten und eine Sequenzlänge von 64. Es wurden 50% der Label für den Unsupervised-Learning-Anteil entfernt.

2 kompatibel. Es werden zusätzliche Anpassungen benötigt, um diese Funktionalität zu ermöglichen, weshalb für diese Modelle keine Ergebnisse vorliegen. Mit lediglich 50% der annotierten Trainingsdaten und einem von vier Epochs, führt das Feintuning, verglichen zum vorherigen Versuchsaufbau, lediglich ein Achtel der Trainingsschritte aus. Ein Trainingsschritt via GAN ist jedoch rechenaufwendiger, als ein Trainingsschritt mit Supervised-Learning. Dies folgt aus der Notwendigkeit, sowohl den Generator, als auch den Discriminator per Backpropagation zu trainieren. Das Ergebnis des GAN-BERT-Papers ist, dass GANs in NLP-Multiklassenaufgaben bessere Ergebnisse erzielen können. Dies wird mit der höheren Lernfähigkeit mit weniger Trainingsdaten in dieser Trainingsmethode begründet. [114] Der Versuchsaufbau dieser Masterthesis bestätigt eine gute Lernfähigkeit von GANs mit wenigen annotierten Trainingsdaten, auch in einer binären Textklassifizierung.

Im dritten Versuchsaufbau dieser Masterthesis ändert sich zwischen den Trainings der Anteil annotierter Daten. Evaluiert werden die drei Modelle, die in den Versuchen 7.1 und 7.2 die höchsten Genauigkeitsmetriken erzielen. Ziel ist die Schaffung einer Übersicht zur Entwicklung der Modellgenauigkeit mit verschiedenen Anteilen annotierter Daten. Das Ergebnis dieses Versuchs ist in Tabelle 7.3 dargestellt.

Auffällig ist, dass der MCC-Score bis einschließlich einem Anteil von 25% annotierten Kommentaren sehr instabil ist. Bei einem Anteil annotierter Trainingsdaten unter fünf Prozent ist das Ergebnis mit einem MCC von durchschnittlich circa Null, nicht präziser als ein Münzwurf. Die Aufgabe der Toxizitätsbestimmung ist sehr komplex, weshalb ein so kleiner Anteil an Trainingsdaten kein Verständis für Toxizität abbilden kann. Ab einem Anteil von 50% annotierter Trainingsdaten stabilisieren sich jedoch alle drei evaluierten Modelle in ihrer Genauigkeit. Werden den Modellen mehr als 50% der Trainingsdaten annotiert, bietet das Training weiterhin inkrementelle Verbesserungen.

Um diese Werte mit dem Lernen ohne eines GANs vergleichbar zu machen, befindet sich in der Tabelle 7.3 auch das Ergebnis mit einem Training ohne GAN. Die Hyperparameter für diesen Vergleich sind ebenfalls ein Epoch und eine Learning-Rate von 5e-5. Wird nur ein Epoch verwendet, erzielt ein GAN demnach ein um zwischen 0,44 und 2,5

Anteil annotiert	dbmdz cased	dbmdz uncased	deepset cased
1%	**0,93%**	-3,76%	-1,03%
5%	1,23%	**10,46%**	0,00%
10%	**9,46%**	-0,83%	2,40%
25%	**34,34%**	2,50%	8,59%
50%	36,53%	**42,18%**	40,42%
75%	**46,74%**	43,95%	**46,77%**
90%	46,80%	**50,59%**	46,84%
100%	**48,53%**	49,46%	48,28%
ohne GAN	46,02%	48,91%	47,84%

Tabelle 7.3: MCC-Score der GermEval 2018 Shared Task 1 mit einem GAN-Epoch, Batchgröße 32, 5e-5 Learning-Rate und einer Sequenzlänge von 64. Das Warmup beträgt 10% der annotierten Daten. Ein Feintuning mit 100% annotierten Trainingsdaten wurde mit gleichen Hyperparametern ohne GAN durchgeführt.

Prozentpunkte besseres Ergebnis im MCC-Score, als es ohne GAN möglich ist. Das Ausführen weiterer GAN-Epochs mit ansonsten gleichen Hyperparametern führt jedoch nicht zu weiteren Genauigkeitsgewinnen.

7.3 Vorverarbeitung

Um die Widerstandsfähigkeit der Klassifizierung gegen unnatürliche Sprache zu härten, stehen mehrere Möglichkeiten zur Verfügung. Im sogenannten *“Leetspeak”* (auch *“1337”* genannt) werden Buchstaben mit ähnlich aussehenden Ziffern ausgetauscht. Leetspeak kann auch in stärkeren Ausprägungen auftreten, bei welchen Buchstaben mit Sonderzeichen oder Symbolen fremder Sprachen getauscht werden. Dies hat zur Wirkung, dass die hierbei entsehenden Wörter von Menschen gelesen werden können, aber nicht in Wörterbüchern gefunden werden können. Beispielsweise kann das Wort *“Wikipedia”* in Leetspeak mit *“w!k!p3d!4”* ersetzt werden. [130, S. 1]

Im Paper *“Text Processing Like Humans Do: Visually Attacking and Shielding NLP Systems”* von Eger et al. 2019 wird eine Methode beschrieben, welche die Eingangsdaten visuell untersucht und das Zeichen in den nahegelegensten Buchstaben umwandelt. [130] Da dieser Ansatz in seiner Implementierung einen hohen Aufwand birgt, wird stattdessen ein weniger komplexes *“Unleet”* in Python implementiert. Die Ziffern *“0123456789”* werden respektive mit *“olzeasgtbg”* ersetzt. Des Weiteren werden Symbole, wie *“$€!(αβγεζηθϑικνξπϖρσςτυϕφχψω”* respektive mit *“seicabrecnoikvenpoctvqxyw”* ersetzt.

```
#!/usr/bin/python
in_leet_list=["0","1","2","3","4","5","6","7","8",...]
```

```
un_leet_list=["o","l","z","e","a","s","g","t","b",...]

def unleet(input_comment):
  un1337 = input_comment
  for i, j in enumerate(in_leet_list):
    un1337 = un1337.replace(j,un_leet_list[i])
  return un1337

print(unleet("1337"))
```

```
>> leet
```

In den GermEval-Datensätzen ist Leetspeak nicht vertreten, weshalb sich mit dieser Implementierung die Genauigkeitswerte nicht verändern. Was jedoch für einen Verarbeitungsoverhead sorgt, ist die Implementierung von " `|LBR|` " zur Signalisierung eines Zeilenumbruchs. Wird dieser Teil des Inputs entfernt, werden weniger Token für die Verarbeitung durch das ANN benötigt. Hierdurch werden bei Kommentaren mit Zeilenumbrüchen durch die maximale Sequenzlänge, effektiv mehr Token mit Kommentarinhalt verarbeitet.

```
#!/usr/bin/python
def lbr_remove(input_comment):
  input_comment = input_comment.replace(". |LBR|",".")
  input_comment = input_comment.replace(" |LBR|",".")
  return input_comment

print(lbr_remove("Guten Morgen aus Bordesholm. |LBR| Auf dem Weg
    nach Schleswig"))
```

```
>> Guten Morgen aus Bordesholm. Auf dem Weg nach Schleswig
```

Ferner kann der zu bewertende Kommentar gekürzt werden, indem die User Mentions entfernt werden. Diese ist eine der Methoden, mit dem das Team, mit der höchsten Genauigkeit in der GermEval 2018 Shared Task 1, das Feintuning optimierte. [8, S. 239] Diese Funktionalität ist bereits in dem BERT-Tokenisierer implementiert und kann als Argument per `strip_handles=True` aktiviert werden. Eine ebenfalls im BERT-Tokenisierer enthaltene Funktion heißt `reduceLen=True`. Ist diese aktiviert, werden gleiche, aufeinanderfolgende Buchstaben auf eine maximale Länge von drei gekürzt. So wird das Wort *"cooooool"* zu *"coool"* verkürzt und nimmt dementsprechend weniger Token in Anspruch. [16, S. 46]

Während der Evaluierung der vorgestellten Vorverarbeitungsmethoden konnte kein konsistenter Genauigkeitsgewinn festgestellt werden. Für einen Einsatz mit anderen Datensätzen können diese Vorverarbeitungen der Eingangsdaten jedoch in manchen Fällen Verbesserungen erzielen und das Modell gegen Angriffe härten.

7.4 Hyperparameteranpassung

Einen großen Einfluss auf die Genauigkeitswerte des Modells wirken die Hyperparameter aus. Während für die Modelloptimierung mittels einer Hyperparameteranpassung für BERT-Modelle bereits viele Publikationen existieren, ist die Datenlage zu dem im Juli 2020 erschienenen GAN-BERT sehr gering. Daher wird in diesem Kapitel eine Hyperparameteroptimierung für GAN-BERT vorgenommen, indem einzelne oder zusammenhängende Hyperparameter gegenüber dem GAN-BERT-Paper angepasst werden. [114]

Die Learning-Rate und die Anzahl der Epochs stehen im direkten Zusammenhang zueinander. Wenn ein Modell mit mehreren Epochs trainiert wird, benötigt es eine geringere Learning-Rate, um vergleichbare Ergebnisse zu erzielen. Dies bestätigt sich in dem Versuchsaufbau, bei welchem mit bis zu vier GAN-Epochs die Learning-Rate variiert wird, um den höchstmöglichen MCC-Score im Testdatensatz zu erreichen. In der Tabelle 7.4 befinden sich die Ergebnisse dieses Versuchs. Die durchschnittlich besten Modelle erreichen in diesem Versuch die Durchgänge mit 2 GAN-Epochs und einer Learning-Rate zwischen 4e-5 und 5e-5. Mit dieser Hyperparameteranpassung kann der MCC-Score gegenüber dem vorherigen Versuchsaufbau um etwa zehn Prozentpunkte erhöht werden.

GAN-Epochs	Learning-Rate	dbmdz cased	dbmdz uncased	deepset cased
1	2e-5	36,21%	39,68%	35,34%
	3e-5	40,99%	**43,67%**	38,84%
	4e-5	**43,08%**	42,37%	40,28%
	5e-5	36,53%	42,18%	**40,42%**
	6e-5	39,92%	30,64%	39,77%
2	2e-5	44,90%	46,74%	46,00%
	3e-5	44,39%	48,57%	45,80%
	4e-5	**46,60%**	46,07%	**47,12%**
	5e-5	44,10%	**52,08%**	44,78%
	6e-5	42,15%	46,15%	40,93%
3	9e-6	47,11%	44,45%	41,34%
	1e-5	**48,48%**	**46,81%**	41,61%
	2e-5	48,07%	44,85%	**44,85%**
	3e-5	46,74%	41,54%	41,79%
	4e-5	46,59%	47,92%	39,18%
4	8e-6	41,07%	45,09%	42,56%
	9e-6	44,15%	43,98%	**43,20%**
	1e-5	44,21%	43,09%	42,28%
	2e-5	**47,78%**	**47,86%**	38,77%
	3e-5	42,53%	44,70%	37,70%

Tabelle 7.4: MCC-Score der GermEval 2018 Shared Task 1, Batchgröße 32, Sequenzlänge von 64, 10% der annotierten Daten als Warmup und 50% weniger Annotationen.

Das `bert-base-german-dbmdz-uncased`-Modell liegt somit nach lediglich zwei GAN-Epochs und 50% der annotierten Daten mit einem MCC von 52,08% nur 2,07 Prozentpunkte hinter dem Supervised-Learning Modell, welches vier Epochs und 100% der annotierten Daten verwendet. Das Supervised-Learning Modell erreicht, wie in Tablle 7.1 dargestellt, einen MCC von 54,15%. Weitere Hyperparameteranpassungen werden anhand eines GAN-Epochs evaluiert. Somit wird die benötigte Zeit für das Feintuning halbiert. Die verbesserte Lernfähigkeit kann anhand eines einzigen Epochs dargestellt werden, weshalb nicht von einer Verfälschung des Ergebnisses auszugehen ist.

Batch-Größe	dbmdz cased	dbmdz uncased	deepset cased
8	34,24%	42,09%	40,10%
16	36,93%	40,51%	**43,13%**
32	**43,08%**	42,37%	40,28%
64	14,59%	**45,54%**	33,38%

Tabelle 7.5: MCC-Score der GermEval 2018 Shared Task 1, ein GAN-Epoch, Learning-Rate 4e-5, Sequenzlänge von 64, 10% der annotierten Daten als Warmup und 50% weniger Annotationen.

Die Ausführungsgeschwindigkeit des Feintunings steht im direkten Zusammenhang mit der Batchgröße. Für das englischsprachige BERT-Modell empfehlen die Autoren, je nach Aufgabe, eine Batchgröße von 16 oder 32. [110, S. 6] Je mehr Batches gleichzeitig verarbeitet werden, desto höher ist der Speicherbedarf der GPU oder TPU. In Google Colab Pro steht für das Feintuning der Versuche eine NVidia Tesla P100 mit 16GB HBM2-Speicher zur Verfügung. Diese GPU ermöglicht eine Ausführung von Batchgrößen bis zu 64, wenn die Sequenzlänge 200 beträgt. Die Ergebnisse für diesen Versuch sind in Tabelle 7.5 dargestellt. Je nach Modell ist demnach eine Batch-Größe zwischen 16 und 64 empfehlenswert. Für weitere Evaluierungen der Modelle, wird eine Batchgröße von 32 verwendet, da diese für alle geprüften Modelle stabile Ergebnisse liefert.

Neben der Anzahl der Epochs steht auch die Sequenzlänge im direkten Zusammenhang mit der Ausführungszeit. Der längste Kommentar im Trainingsdatensatz verfügt über eine Länge von 195 Token. In Abbildung 7.1 ist logarithmisch dargestellt, dass die Anzahl der Kommentare mit steigender Tokenlänge stetig sinkt. Dies kann eine Erklärung dafür sein, dass die Ergebnisse des Versuchsaufbaus aus Tabelle 7.6 keine klare Entwicklung zeigen. Während bei den Modellen der dbmdz die Genauigkeitswerte stark schwanken, scheint das Modell von deepset von der höheren Sequenzlänge zu profitieren. Um längere Kommentare besser klassifizieren zu können, wird für weitere Evaluierungen im Rahmen dieser Masterthesis eine Sequenzlänge von 128 statt 64 verwendet.

Wie verändern sich die Ergebnisse, wenn alle optimierten Hyperparameter gleichzeitig auf die Modelle angewandt werden? Hierfür findet ein Vergleich mit dem Versuchsaufbau aus Tabelle 7.2 statt. In beiden Versuchen werden 50% der Label entfernt und

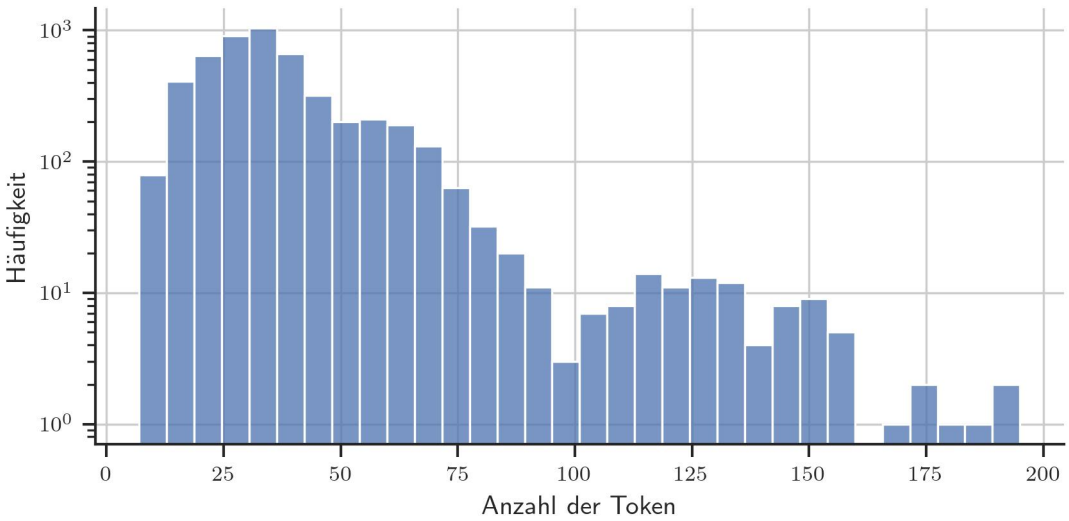

Abbildung 7.1: Verteilung der Tokenlängen der Kommentare im GermEval-Trainingsdatensatz 2018

Sequenz-länge	dbmdz cased	dbmdz uncased	deepset cased
64	**43,08%**	42,37%	40,28%
96	38,34%	45,71%	35,75%
128	39,93%	**49,56%**	41,01%
192	36,04%	43,55%	44,36%
256	41,59%	48,97%	**45,28%**

Tabelle 7.6: MCC-Score der GermEval 2018 Shared Task, ein GAN-Epoch, Learning-Rate 4e-5, Batchgröße 32, 10% der annotierten Daten als Warmup und 50% weniger Annotationen.

die Warmups betragen 10% der annotierten Trainingsdaten. Statt einem GAN-Epoch werden zwei verwendet. Auch die Learning-Rate beträgt 4e-5 statt 5e-5. Die maximale Sequenzlänge beträgt für diesen Versuch 128. Die Ergebnisse befinden sich in Tabelle 7.7. Die Modelle mit Orthografie sind den Modellen ohne Orthografie vor und nach der Hyperparameteroptimierung in den gewählten Genauigkeitsmetriken überlegen.

7.5 Übertragen der Ergebnisse auf GermEval 2021

Wie übertragbar sind diese Ergebnisse auf andere Datensätze, wie der aktuellen GermEval-Herausforderung? Hierfür werden im Versuchsaufbau 7.8 die in Versuchsaufbau 7.1 und 7.7 trainierten Modelle ohne weitere Anpassungen anhand des Testdatensatzes von GermEval 2021 beurteilt. Die Baseline-Performance liegt bei einem F1-Score von 38,62% mit einem Mehrheitsklassen-Klassifikator.

Modell	Accuracy	Precision	Recall	F1	MCC	Δ MCC 7.2
bert-base-german-cased	76.80%	65.36%	66.78%	**74.22%**	48.45%	8,08%
bert-base-german-dbmdz-cased	75.21%	60.85%	**74.80%**	73.61%	48.20%	1,04%
bert-base-german-dbmdz-uncased	**79.30%**	**76.39%**	56.15%	75.04%	**51.76%**	0,80%
dbmdz/bert-base-german-europeana-cased	75.91%	75.00%	43.16%	69.19%	42.79%	13,23%
dbmdz/bert-base-german-europeana-uncased	75.29%	63.19%	64.52%	72.54%	45.09%	12,63%

Tabelle 7.7: Genauigkeitsmetriken der GermEval 2018 Shared Task 1 mit zwei GAN-Epochs, Batchgröße 32, 4e-5 Learning-Rate, 10% Warmup der annotierten Daten und eine Sequenzlänge von 128. Es wurden 50% der Label für den Unsupervised-Learning-Anteil entfernt. Vergleich zu Versuchsaufbau 7.2

Modell	Accuracy	Precision	Recall	F1	MCC
bert-base-german-cased	**67.81%**	68.55%	**24.29%**	**57.18%**	**25.34%**
bert-base-german-dbmdz-cased	67.08%	**69.70%**	19.71%	54.56%	23.12%
bert-base-german-dbmdz-uncased	66.98%	67.92%	20.57%	54.90%	22.71%
bert-base-german-cased (GAN)	68.75%	65.71%	33.33%	**61.26%**	28.47%
bert-base-german-dbmdz-cased (GAN)	66.70%	57.89%	**38.26%**	60.99%	24.47%
bert-base-german-dbmdz-uncased (GAN)	**68.86%**	**79.17%**	22.03%	57.02%	**29.51%**

Tabelle 7.8: Genauigkeitsmetriken der GermEval 2021 Teilaufgabe 1 nach Parameteroptimierung, ohne Trainingsdaten der 2021-Challenge.

Die besten Ergebnisse erzielen die BERT-Modelle von deepset und der dbmdz. Obwohl die GAN-Modelle weniger Zeit für das Training benötigen und nur die hälfte der annotierten Datensätze im Feintuning zur Verfügung haben, erreichen diese Modelle in allen Genauigkeitsmetriken bessere Ergebnisse, als die via Supervised-Learning angepassten Modelle.

Die GermEval-Daten von 2018 und 2021 wurden nicht nur zu unterschiedlichen Zeitpunkten gesammelt und bewertet, sondern auch von unterschiedlichen Plattformen. Um dies anzugleichen, findet in Versuch 7.9 ein Feintuning mit den Daten von GermEval 2021 statt. Durch die gleiche Herkunft der Daten, kann die Genauigkeit auf den Testdatensatz erhöht werden. Ohne GAN beträgt der Genauigkeitsgewinn im Macro-F1-Score zwischen sieben und zehn Prozentpunkte. Mit GAN fällt der Genauigkeitsgewinn mit etwa zwei bis fünf Prozentpunkten geringer aus. Das Modell der dbmdz mit Orthografie büßt sogar 12,54 Prozentpunkte im Macro-F1-Score ein.

Als letzten Versuchsaufbau dient der in Tabelle 7.10 dargestellte Versuch. In diesem wird das Modell robuster trainiert, indem alle zur Verfügung stehenden Trainingsdaten der GermEval-Ausgaben 2018, 2019 und 2021 genutzt werden. Hierdurch soll die Einsatzfähigkeit des Modells über mehrere Plattformen und Beobachtungszeiträume hinaus steigen.

Modell	Accuracy	Precision	Recall	F1	MCC
Perspective API	69.60%	66.15%	36.86%	62.98%	30.72%
bert-base-german-cased	68.96%	61.60%	**44.00%**	64.33%	30.47%
bert-base-german-dbmdz-cased	68.75%	61.67%	42.29%	63.76%	29.73%
bert-base-german-dbmdz-uncased	**70.52%**	**68.34%**	38.86%	**64.43%**	**33.45%**
bert-base-german-cased (GAN)	64.55%	51.81%	**66.38%**	**63.71%**	**28.88%**
bert-base-german-dbmdz-cased (GAN)	65.52%	**75.51%**	10.72%	48.45%	18.73%
bert-base-german-dbmdz-uncased (GAN)	**68.64%**	63.78%	36.23%	62.04%	28.48%

Tabelle 7.9: Genauigkeitsmetriken der GermEval 2021 Teilaufgabe 1 mit Trainingsdaten der 2021-Challenge. 4 Epochs oder 2 GAN-Epochs mit 50% weniger Annotationen.

Die Evaluierung findet anhand der Daten von GermEval 2021 statt. Ein vierter Epoch erhöht die Genauigkeiten gegenüber drei Epochs nicht, weshalb drei Epochs verwendet werden. Dies ist eine Folge der hohen Datenmenge in diesem Versuchsaufbau. Die Trainingsdauer für einen Epoch ist vierfach so lang, da etwa 20.000 statt etwa 5.000 Trainingsdaten zur Verfügung stehen. Die Verwendung von zwei GAN-Epochs bringt bessere Ergebnisse als die Verwendung eines GAN-Epochs. Trotz der höheren Datenmenge steigen die Genauigkeitsmetriken für diesen Versuchsaufbau nicht. Ein solcher Trainingsansatz ist für den Einsatz über die Challenge hinaus dennoch einem quantitativ geringer trainiertem Modell vorzuziehen.

Modell	Accuracy	Precision	Recall	F1	MCC
bert-base-german-cased	66.46%	58.71%	33.71%	59.64%	23.29%
bert-base-german-dbmdz-cased	**70.10%**	**69.54%**	**34.57%**	**62.75%**	**31.95%**
bert-base-german-dbmdz-uncased	67.71%	64.81%	30.00%	59.53%	26.14%
bert-base-german-cased (GAN)	68.86%	63.33%	**38.55%**	**62.86%**	**29.27%**
bert-base-german-dbmdz-cased (GAN)	**68.97%**	73.17%	26.09%	58.86%	29.11%
bert-base-german-dbmdz-uncased (GAN)	68.10%	**79.52%**	19.13%	55.06%	27.46%

Tabelle 7.10: Genauigkeitsmetriken der GermEval 2021 Teilaufgabe 1 mit Trainingsdaten von GermEval 2018, 2019 und 2021. 3 Epochs oder 1 GAN-Epoch mit 50% weniger Annotationen.

8 Diskussion

Die automatisierte Erkennung von Toxizität ist eine komplexe Aufgabe. Um diese besser zu bewältigen, werden immer größere Modelle und Trainingsdaten benötigt. Während reine ML-Modelle wenige annotierte Daten für das Training benötigen, können DL-Modelle Gebrauch von großen Datensätzen machen, die nicht annotiert sind. Die Transformer-Architektur profitiert sowohl von Unsupervised-Learning im Pre-Training, als auch von Supervised-Learning im Feintuning. Dieses Semi-Supervised-Learning in der Transformer-Architektur ist ein Grund für den Erfolg der darauf basierenden Modelle.

Die Genauigkeitswerte für englischsprachige Modelle in Benchmarks, wie SuperGLUE, sind bemerkenswert. Da weniger Trainingsdaten für deutschsprachige Modelle zur Verfügung stehen und weniger qualitativ trainierte Modelle existieren, besteht eine Kluft zwischen deutschsprachigen und englischsprachigen Modellen, die noch nicht durch die Komplexität multilingualer trainierter Modelle geschlossen werden kann. Sowohl das Pre-Training, als auch das Feintuning deutschsprachiger Modelle verfügt über durchschnittlich weniger Trainingsdaten und verfügbare Rechenzeit gegenüber den englischsprachigen Modellen.

Während für das Pre-Training, mit einem größerem Korpus und vielfach mehr Trainingsschritten, kostenintensive Lösungsansätze für eine bessere Modellperformance möglich sind, kann das Feintuning durch den Einsatz von GANs optimiert werden. Die Optimierung besteht hierbei in der Reduktion der benötigten annotierten Daten und den benötigten Epochs, was geringere Kosten im Feintuning verursacht. Untersuchungen für die Nutzung von GANs in Textklassifizierungsaufgaben existieren bisher kaum. Im deutschen Sprachraum konnte im Rahmen dieser Masterthesis keine Arbeit mit ähnlicher Herangehensweise ausfindig gemacht werden.

Die Empfehlung des Autors, GANs für Multiklassenaufgaben einzusetzen, wurde nicht geprüft. [114] Stattdessen wurde evaluiert, ob GANs bei binärer Textklassifizierung mit wenigen Trainingsdaten ebenfalls gute Erfolge erzielen. Dies konnte in dieser Arbeit validiert werden. Kostenreduzierungen durch kleinere Datensätze mit GANs und eine geringere Anzahl von benötigten Trainingsschritten stehen jedoch einem hohen Experimentieraufwand gegenüber.

Wird ein klassisches Supervised-Learning im Feintuning für Toxizitätsklassifizierungsaufgaben verwendet, können Genauigkeitswerte über denen der Google Jigsaw Perspective API erreicht werden. Besonders gute MCC-Scores erreichen die BERT-Modelle der dbmdz und von deepset.

Durch das robustere Training verbessern sich die Genauigkeitsmetriken einzelner Testdatensätze nicht. Jedoch kann das Modell mehr Sachverhalte aus verschiedenen Zeiträumen aufnehmen und somit vielfältiger eingesetzt werden. Dies ist auch ein Vorteil von der Perspective API. Die enorme Datenmenge, die Google zur Verfügung steht, erlaubt es ein Modell zu kreiern, welches in vielen Aufgaben gute Ergebnisse erzielt.

Eine vollständige Hyperparameteranpassung für GermEval 2021 Shared Task 1 wurde in dieser Masterthesis nicht vorgenommen. Während der erarbeitete Trainingsansatz für ANNs vergleichbare Ergebnisse zu den besten Ergebnissen von GermEval 2018 erzielt, liegt die Genauigkeit unter dem Median der Modelle von GermEval 2021, welcher bei einem F1-Score von 66,85% liegt. In dieser Arbeit trainierte Modelle verfügen bei diesem Datensatz über einen F1-Score von maximal 64,43%. Die verwendeten Modelle haben das Potential höhere Genauigkeitsmetriken auf den Testdatensatz zu erreichen, wie beispielsweise die Einreichungen der FH Aachen beweisen. Diese verwenden unter anderem das BERT-Modell von deepset und erreichen mit diesem in der Modellerkundung einen F1-Score von bis zu 72%. [40, S. 109]

Weiterhin kann das Semi-Supervised-Learning durch GANs besser genutzt werden, indem während des Trainings keine Label entfernt werden, sondern ähnliche Datensätze ausfindig gemacht werden, die über keine Annotationen verfügen. Alle in dieser Arbeit via GAN trainierten Modelle wurden mit 50% Annotationen trainiert, indem die Hälfte der Label entfernt wurden. Bei weiteren Forschungen zu GANs in binärer Textklassifizierung kann der Datensatz beispielsweise verdoppelt werden, indem Daten von den zu bewertenden Plattformen gecrawlt werden und in den Trainingsdatensatz integriert werden.

Da die verwendeten GANs vielversprechende Ergebnisse in den gewählten Versuchen erzielen, bieten diese eine gute Grundlage für weitere Forschung auf dem Gebiet des NLP. Besonders in Multiklassenaufgaben mit wenigen verfügbaren annotierten Trainingsdaten sind gute Genauigkeitswerte zu erwarten, sollten die Ergebnisse des GAN-BERT-Papers in den deutschen Sprachraum übertragbar sein. [114]

Auch scheint das Potential von Modellen mit Orthografie noch nicht ausgeschöpft zu sein. Mit ausreichenden Trainingsressourcen und einem größeren Vokabular sollten diese Modelle in der Lage sein, mit möglicher Groß- und Kleinschreibung in den Token ein größeres Wissen abzubilden. Diese Modelle erreichen zwar bessere Genauigkeitswerte in der GermEval-Herausforderung von 2021, liegen jedoch bei dem Testdatensatz 2018 hinter den Modellen ohne Orthografie. Der Unterschied zwischen den Plattformen könnte hierbei eine Rolle spielen. Während 2018 Twitter verwendet wurde, stammen die Daten 2021 von Facebook. Für die Beantwortung dieser Frage müssen die Plattformen in ihrer vorkommenden Sprache gegenübergestellt werden. Dies könnte in Zukunft die Vorverarbeitung von Daten plattformspezifisch gestalten.

Literatur

[1] Bundeskriminalamt, *Bundeslagebild Cybercrime 2020*. Bundeskriminalamt, April 2021.

[2] ARD/ZDF-Forschungskommission. „Internetnutzer* in Deutschland 2016 bis 2020 - Soziodemografie.“ (5. August 2021), Adresse: https://www.ard-zdf-onlinestudie.de/onlinenutzung/internetnutzer/in-mio/ (besucht am 5. August 2021).

[3] S. Wünsch. „BGH: Internet ist ein Grundrecht,“ Deutsche Welle. (15. Februar 2013), Adresse: https://p.dw.com/p/17RO6 (besucht am 5. August 2021).

[4] C. Schemer, N. Jackob, O. Quiring, T. Schultz, M. Ziegele und V. Granow. „Wahrnehmung von Fake News und Hasskommentaren (2017).“ (15. Februar 2018), Adresse: http://www.uni-mainz.de/presse/aktuell/Dateien/02_publizistik_medienvertrauen__2017_grafiken.pdf.

[5] K. Marx, „Warum automatische Verfahren bei der Detektion von Hate Speech nur die halbe Miete sind,“ in *Cyberkriminologie*, Springer Fachmedien Wiesbaden, 2020, S. 707–725. DOI: 10.1007/978-3-658-28507-4_27.

[6] InternetLiveStats.com. „Twitter Usage Statistics,“ W3C. (5. August 2021), Adresse: https://www.internetlivestats.com/twitter-statistics/ (besucht am 5. August 2021).

[7] J. Wright. „Updates on our efforts to make YouTube a more inclusive platform,“ Google. (3. Dezember 2020), Adresse: https://blog.youtube/news-and-events/make-youtube-more-inclusive-platform/ (besucht am 18. September 2021).

[8] I. Vogel, R. Regev und M. Steinebach, „Automatisierte Analyse Radikaler Inhalte im Internet,“ de, 2019. DOI: 10.18420/INF2019_27.

[9] H. Gleiß. „Deep Learning zur Klassifizierung von Hate Speech: Zwischenergebnisse des NOHATE Forschungsprojekts,“ Das NETTZ. (12. Januar 2019), Adresse: https://www.das-nettz.de/deep-learning-zur-klassifizierung-von-hate-speech-zwischenergebnisse-des-nohate-forschungsprojekts (besucht am 7. August 2021).

[10] pbe. „Facebooks Forscher zweifeln an Fähigkeiten der Hass-Erkennung,“ DER SPIEGEL. (18. Oktober 2021), Adresse: https://www.spiegel.de/netzwelt/web/kuenstliche-intelligenz-facebook-zweifelt-an-faehigkeiten-der-hass-erkennung-a-c322d904-da87-4e3b-80b8-ba427186c64d (besucht am 18. Oktober 2021).

[11] G. Rosen. „Hate Speech Prevalence Has Dropped by Almost 50% on Facebook," Facebook. (17. Oktober 2021), Adresse: https://about.fb.com/news/2021/10/hate-speech-prevalence-dropped-facebook/ (besucht am 19. Oktober 2021).

[12] Texterclub e. K. „Einführung in die Morphologie," SGV Verlag. (4. August 2021), Adresse: https://www.texterclub.de/mehrdeutige-woerter/ (besucht am 4. August 2021).

[13] B. Talafha, M. E. Za'ter, S. Suleiman, M. Al-Ayyoub und M. N. Al-Kabi, „sarcasm detection and quantification in arabic tweets," 3. August 2021. arXiv: 2108.01425 [cs.CL].

[14] H. Chang. „Visualizing ELMo Contextual Vectors," Medium. (15. April 2019), Adresse: https://towardsdatascience.com/visualizing-elmo-contextual-vectors-94168768fdaa (besucht am 7. August 2021).

[15] Bundeszentrale für politische Bildung. „Was ist Hate Speech?" (12. Juli 2017), Adresse: https://www.bpb.de/252396/was-ist-hate-speech (besucht am 24. Juli 2021).

[16] J. Ruppendorfer, M. Siegel und M. Wiegand, *konvens 2018 - GermEval Proceedings*. Verlag der Österreichischen Akademie der Wissenschaften, 2018. DOI: 10.1553/0x003a105d.

[17] R. Pandarachalil, S. Sendhilkumar und G. S. Mahalakshmi, „Twitter Sentiment Analysis for Large-Scale Data: An Unsupervised Approach," *Cognitive Computation*, Jg. 7, Nr. 2, S. 254–262, November 2014. DOI: 10.1007/s12559-014-9310-z.

[18] C. J. Kennedy, G. Bacon, A. Sahn und C. von Vacano, „Constructing interval variables via faceted Rasch measurement and multitask deep learning: a hate speech application," 22. September 2020. arXiv: 2009.10277 [cs.CL].

[19] F. Poecze, C. Ebster und C. Strauss, „Let's play on Facebook: using sentiment analysis and social media metrics to measure the success of YouTube gamers' post types," *Personal and Ubiquitous Computing*, Dezember 2019. DOI: 10.1007/s00779-019-01361-7.

[20] P. S. Dandannavar, S. R. Mangalwede und S. B. Deshpande, „Emoticons and Their Effects on Sentiment Analysis of Twitter Data," in *EAI International Conference on Big Data Innovation for Sustainable Cognitive Computing*, Springer International Publishing, Oktober 2019, S. 191–201. DOI: 10.1007/978-3-030-19562-5_19.

[21] M. Spranger und D. Labudde, „Vorhersage von Gruppendynamiken auf der Grundlage von Daten aus Sozialen Netzwerken," in *Cyberkriminologie*, Springer Fachmedien Wiesbaden, 2020, S. 653–683. DOI: 10.1007/978-3-658-28507-4_25.

[22] *Proceedings of the International Conference on Inventive Research in Computing Applications (ICIRCA 2018) : date: July 11-12, 2018*. Piscataway, New Jersey: IEEE, 2018, ISBN: 9781538624562.

[23] J. M. Struß, M. Siegel, J. Ruppendorfer, M. Wiegand und M. Klenner, *konvens 2019 - GermEval Proceedings*. Verlag der Österreichischen Akademie der Wissenschaften, 2019.

[24] N. Vaidya. „5 Natural Language Processing Techniques for Extracting Information," Aureus Analytics. (18. Mai 2020), Adresse: https://blog.aureusanalytics.com/blog/5-natural-language-processing-techniques-for-extracting-information (besucht am 5. August 2021).

[25] The MathWorks, Inc. „Terminology - Deep Learning vs Machine Learning." (2021), Adresse: https://explore.mathworks.com/machine-learning-vs-deep-learning/chapter-1-129M-833I7.html (besucht am 5. August 2021).

[26] D. Chicco und G. Jurman, „The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," Jg. 21, Nr. 1, Januar 2020. DOI: 10.1186/s12864-019-6413-7.

[27] K. P. Shung. „Accuracy, Precision, Recall or F1?" Medium. (15. März 2018), Adresse: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9 (besucht am 14. Oktober 2021).

[28] Hessisches Ministerium des Innern und für Sport. „Was ist Hate Speech bzw. Hassrede?" (2021), Adresse: https://hessengegenhetze.de/fragen-antworten/hate-speech (besucht am 19. Mai 2021).

[29] K. Biermann. „Der Facebook-Chef und seine dunkle Vergangenheit," Der Tagesspiegel. (8. März 2010), Adresse: https://www.tagesspiegel.de/themen/digitalisierung-ki/mark-zuckerberg-der-facebook-chef-und-seine-dunkle-vergangenheit/1715278.html (besucht am 6. August 2021).

[30] Facebook, Inc. „FB Earnings Presentation Q2 2021." (Juli 2021), Adresse: https://s21.q4cdn.com/399680738/files/doc_financials/2021/q2/Q2-2021_Earnings-Presentation.pdf.

[31] Reuters Limited. „YouTube serves up 100 million videos a day online." (16. Juli 2006), Adresse: https://usatoday30.usatoday.com/tech/news/2006-07-16-youtube-views_x.htm (besucht am 6. August 2021).

[32] dpa und jk. „Facebook schließt Instagram-Kauf ab," Heise. (7. September 2012), Adresse: https://www.heise.de/newsticker/meldung/Facebook-schliesst-Instagram-Kauf-ab-1702270.html (besucht am 6. August 2021).

[33] mbö. „Videos mit bis zu 60 Minuten - Instagram fordert YouTube heraus,“ DER SPIEGEL. (21. Juni 2018), Adresse: https://www.spiegel.de/netzwelt/apps/instagram-neue-app-igtv-bietet-jetzt-videos-mit-bis-zu-60-minuten-laenge-a-1214146.html (besucht am 6. August 2021).

[34] Redaktion CHIP. „Snapchat: Sexting-App löscht Videos nicht.“ (22. November 2013), Adresse: https://www.chip.de/news/Snapchat-Sexting-App-loescht-Videos-nicht_138922457.html (besucht am 6. August 2021).

[35] Snap Inc. „Snap Inc. Announces Second Quarter 2021 Financial Results.“ (22. Juli 2021), Adresse: https://investor.snap.com/news/news-details/2021/Snap-Inc.-Announces-Second-Quarter-2021-Financial-Results/default.aspx (besucht am 6. August 2021).

[36] T. Wittenhorst. „Videoplattform TikTok blockiert Inhalte mit Bezug zu Homosexualität,“ Heise. (30. September 2019), Adresse: https://www.heise.de/newsticker/meldung/Videoplattform-TikTok-blockiert-Inhalte-mit-Bezug-zu-Homosexualitaet-4542800.html (besucht am 6. August 2021).

[37] P. Welchering und U. Blumenthal. „Daten von 5,5 Millionen deutschen TikTok-Nutzern waren gefährdet,“ Deutschlandfunk. (8. Januar 2020), Adresse: https://www.deutschlandfunk.de/sicherheitsluecke-daten-von-5-5-millionen-deutschen-tiktok.676.de.html?dram:article_id=467438 (besucht am 6. August 2021).

[38] D. Yu. „TikTok has 50 million daily active users in the US, it says in lawsuit,“ Tech in Asia. (25. August 2020), Adresse: https://www.techinasia.com/tiktok-unveils-50m-dau-92m-mau (besucht am 6. August 2021).

[39] Deutsche Gesellschaft für Sprachwissenschaft. „Konferenz zur Verarbeitung natürlicher Sprache (KONVENS),“ DGfS. (2021), Adresse: https://dgfs.de/de/cl/konvens.html (besucht am 7. August 2021).

[40] J. Risch, A. Stoll, L. Wilms und M. Wiegand, „Overview of the GermEval 2021 Shared Task on the Identification of Toxic, Engaging, and Fact-Claiming Comments,“ in *Proceedings of the GermEval 2021 Shared Task on the Identification of Toxic, Engaging, and Fact-Claiming Comments*, Association for Computational Linguistics, September 2021, S. 1–12. Adresse: https://aclanthology.org/2021.germeval-1.1.

[41] SemEval. „SemEval-2022 - The 16th International Workshop on Semantic Evaluation,“ GitHub. (7. September 2021), Adresse: https://semeval.github.io/SemEval2022/ (besucht am 7. September 2021).

[42] A. Palmer, N. Schneider, N. Schluter, G. Emerson, A. Herbelot und X. Zhu, Hrsg., *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, Online: Association for Computational Linguistics, August 2021. Adresse: https://aclanthology.org/2021.semeval-1.0.

[43] Google LLC. „Research areas.“ (7. September 2021), Adresse: https://research.google/research-areas/ (besucht am 7. September 2021).

[44] ——, „Jigsaw.“ (7. September 2021), Adresse: https://jigsaw.google.com/ (besucht am 7. September 2021).

[45] Jigsaw/Conversation AI. „Jigsaw Multilingual Toxic Comment Classification.“ (23. März 2020), Adresse: https://www.kaggle.com/c/jigsaw-multilingual-toxic-comment-classification/overview/description (besucht am 10. September 2021).

[46] Google LLC. „Using Machine Learning to Reduce Toxicity Online.“ (10. September 2021), Adresse: https://www.perspectiveapi.com/how-it-works/ (besucht am 10. September 2021).

[47] The Conversation AI Github Organization. „Perspective API documentation,“ GitHub. (10. September 2021), Adresse: https://github.com/conversationai/perspectiveapi (besucht am 10. September 2021).

[48] A. Merz. „Google hilft, Forentrolle zu erkennen.“ (23. Februar 2017), Adresse: https://www.golem.de/news/perspective-google-hilft-forentrolle-zu-erkennen-1702-126336.html (besucht am 10. September 2021).

[49] Google LLC. „FAQs.“ (10. September 2021), Adresse: https://developers.perspectiveapi.com/s/about-the-api-faqs (besucht am 10. September 2021).

[50] Alexa Internet, Inc. „Alexa Top 1000 Most Visited Websites.“ (1. Juli 2019), Adresse: https://www.htmlstrip.com/alexa-top-1000-most-visited-websites (besucht am 10. September 2021).

[51] L. Hanu und Unitary team. „Detoxify,“ GitHub. (4. September 2021), Adresse: https://github.com/unitaryai/detoxify (besucht am 12. September 2021).

[52] L. Hanu, J. Thewlis und S. Haco. „How AI Is Learning to Identify Toxic Online Content.“ (8. Februar 2021), Adresse: https://www.scientificamerican.com/article/can-ai-identify-toxic-online-content/ (besucht am 10. September 2021).

[53] Deutschlandfunk Nova. „Fast nur weiß - Rassismus bei KI.“ (6. August 2020), Adresse: https://www.deutschlandfunknova.de/nachrichten/fast-nur-weiss-rassismus-bei-kuenstlicher-intelligenz (besucht am 11. September 2021).

[54] J. Yang. „DeepToxic,“ GitHub. (6. Februar 2019), Adresse: https://github.com/zake7749/DeepToxic (besucht am 12. September 2021).

[55] P. Badjatiya und S. Gupta. „Hate Speech Detection on Twitter,“ GitHub. (8. Dezember 2017), Adresse: https://github.com/pinkeshbadjatiya/twitter-hatespeech/ (besucht am 13. September 2021).

[56] B. Stecanella. „Understanding TF-ID: A Simple Introduction,“ MonkeyLearn. (10. Mai 2019), Adresse: https://monkeylearn.com/blog/what-is-tf-idf/ (besucht am 13. September 2021).

[57] O. Ellrodt und J. Nasr. „Facebook, Google, Twitter agree to delete hate speech in 24 hours: Germany,“ Reuters. (15. Dezember 2015), Adresse: https://www.reuters.com/article/us-germany-internet-idUSKBN0TY27R20151215 (besucht am 14. September 2021).

[58] I. Kottasová und N. Schmidt. „Facebook, Twitter face fines up to $53 million over hate speech,“ CNN. (5. April 2017), Adresse: https://money.cnn.com/2017/04/05/technology/germany-hate-speech/ (besucht am 14. September 2021).

[59] I. Kottasová. „Europe says Twitter is failing to remove hate speech,“ CNN. (1. Juni 2017), Adresse: https://money.cnn.com/2017/06/01/technology/twitter-facebook-hate-speech-europe/index.html (besucht am 14. September 2021).

[60] A. Martin. „Twitter can automatically hide neo-Nazis and white supremacists, but chooses not to,“ Alphr. (19. Oktober 2017), Adresse: https://www.alphr.com/politics/1007424/twitter-can-automatically-hide-neo-nazis-and-white-supremacists-but-chooses-not-to/ (besucht am 14. September 2021).

[61] R. Klar. „Twitter, Facebook to update hate speech moderation,“ The Hill. (12. März 2020), Adresse: https://thehill.com/policy/technology/528567-twitter-facebook-to-update-hate-speech-moderation (besucht am 14. September 2021).

[62] Twitter, Inc. „Richtlinie zu Hass schürendem Verhalten.“ (15. September 2021), Adresse: https://help.twitter.com/de/rules-and-policies/hateful-conduct-policy (besucht am 15. September 2021).

[63] ——, „Twitter Netzwerkdurchsetzungsgesetzbericht Januar - Juni 2021.“ (Juli 2021), Adresse: https://transparency.twitter.com/content/dam/transparency-twitter/netzdg/netzdg-jan-jun-2021.pdf (besucht am 14. September 2021).

[64] mack. „Twitter testet neue Funktion: „Safety Mode” soll Hate Speech eindämmen,“ Heise. (3. September 2021), Adresse: https://www.heise.de/news/Twitter-testet-neue-Funktion-Safety-Mode-soll-Hate-Speech-eindaemmen-6181215.html (besucht am 13. September 2021).

[65] J. Doherty. „Introducing Safety Mode,“ Twitter. (1. September 2021), Adresse: https://blog.twitter.com/en_us/topics/product/2021/introducing-safety-mode (besucht am 13. September 2021).

[66] Twitter, Inc. „About Safety Mode.“ (13. September 2021), Adresse: https://help.twitter.com/en/safety-and-security/safety-mode (besucht am 13. September 2021).

[67] E. Daniel. „Twitter AI: How the platform is embracing artificial intelligence,“ Verdict. (24. Januar 2019), Adresse: https://www.verdict.co.uk/twitter-ai-automation/ (besucht am 15. September 2021).

[68] C. Shu. „https://www.heise.de/newsticker/meldung/Twitter-kauft-Spezialisten-fuer-Videoaufbereitung-3242556.html,“ TechCrunch. (30. Juli 2014), Adresse: https://techcrunch.com/2014/07/29/twitter-acquires-image-search-startup-madbits/ (besucht am 15. September 2021).

[69] Harvard Paulson School. „Machine-learning startup Whetlab acquired by Twitter.“ (17. Juni 2015), Adresse: https://www.seas.harvard.edu/news/2015/06/machine-learning-startup-whetlab-acquired-twitter (besucht am 15. September 2021).

[70] K. Beer. „Twitter kauft Spezialisten für Videoaufbereitung,“ Heise. (21. Juni 2016), Adresse: https://www.heise.de/newsticker/meldung/Twitter-kauft-Spezialisten-fuer-Videoaufbereitung-3242556.html (besucht am 15. September 2021).

[71] G. Panetta. „Twitter reportedly won’t use an algorithm to crack down on white supremacists because some GOP politicians could end up getting barred too,“ Business Insider. (25. April 2019), Adresse: https://www.businessinsider.com/twitter-algorithm-crackdown-white-supremacy-gop-politicians-report-2019-4 (besucht am 18. Oktober 2021).

[72] A. Carlsen und F. Haque. „What Does Facebook Consider Hate Speech? Take Our Quiz,“ The New York Times. (13. Oktober 2017), Adresse: https://www.nytimes.com/interactive/2017/10/13/technology/facebook-hate-speech-quiz.html (besucht am 15. September 2021).

[73] Center for Countering Digital Hate. „Failure to protect - How tech giants fail to act on user reports of anisemitism.“ (August 2021), Adresse: https://252f2edd-1c8b-49f5-9bb2-cb57bb47e4ba.filesusr.com/ugd/f4d9b9_cac47c87633247869bda54fb35399668.pdf (besucht am 15. September 2021).

[74] Facebook, Inc. „Hassrede - Richtliniendetails.“ (16. Dezember 2019), Adresse: https://transparency.fb.com/de-de/policies/community-standards/hate-speech/ (besucht am 16. September 2021).

[75] ——, „Hassrede - Richtliniendetails.“ (11. August 2020), Adresse: https://transparency.fb.com/de-de/policies/community-standards/hate-speech/ (besucht am 16. September 2021).

[76] J. Eigendorf. „Info-Date am Mittag: Malu Dreyer für Corona-Testpflicht bei Einreise und Niederlage für Facebook wegen Löschung von Hate-Speech," SWR. (29. Juli 2021), Adresse: https://www.swr.de/swraktuell/radio/info-date-am-mittag-malu-dreyer-fuer-corona-testpflicht-bei-einreise-und-niederlage-fuer-facebook-wegen-loeschung-von-hate-speech-100.html (besucht am 16. September 2021).

[77] Facebook, Inc. „Community Standards Enforcement Report, November 2019 Edition." (13. November 2019), Adresse: https://about.fb.com/news/2019/11/community-standards-enforcement-report-nov-2019/ (besucht am 16. September 2021).

[78] ——, „AI advances to better detect hate speech." (12. Mai 2020), Adresse: https://ai.facebook.com/blog/ai-advances-to-better-detect-hate-speech/ (besucht am 16. September 2021).

[79] D. Kiela, H. Firooz, A. Mohan, V. Goswami, A. Singh, P. Ringshia und D. Testuggine, „The Hateful Memes Challenge: Detecting Hate Speech in Multimodal Memes," 10. Mai 2020. arXiv: 2005.04790 [cs.AI].

[80] A. Sankin. „YouTube Said It Was Getting Serious About Hate Speech. Why Is It Still Full of Extremists?" Gizmodo. (27. Juli 2019), Adresse: https://gizmodo.com/youtube-said-it-was-getting-serious-about-hate-speech-1836596239 (besucht am 17. September 2021).

[81] S. Wojcicki. „YouTube at 15: My personal journey and the road ahead," Google LLC. (14. Februar 2020), Adresse: https://blog.youtube/news-and-events/youtube-at-15-my-personal-journey/ (besucht am 17. September 2021).

[82] G. D. Vynck. „YouTube says it's getting better at taking down videos that break its rules. They still number in the millions.," The Washington Post. (6. April 2021), Adresse: https://www.washingtonpost.com/technology/2021/04/06/youtube-video-ban-metric/ (besucht am 17. September 2021).

[83] A. Hutchinson. „YouTube Outlines New Measures to Combat Hate Speech on its Platform," Industry Dive. (3. Dezember 2020), Adresse: https://www.socialmediatoday.com/news/youtube-outlines-new-measures-to-combat-hate-speech-on-its-platform/591603/ (besucht am 18. September 2021).

[84] Google LLC. „Google Transparenzbericht - Wichtige Richtlinien." (18. September 2021), Adresse: https://transparencyreport.google.com/youtube-policy/featured-policies/hate-speech (besucht am 18. September 2021).

[85] S. Haykin, *Neural networks: a comprehensive foundation*. Delhi: Pearson Education, 1999, ISBN: 8178083000.

[86] K. O'Shea und R. Nash, „An Introduction to Convolutional Neural Networks," 26. November 2015. arXiv: 1511.08458 [cs.NE].

[87] IBM Cloud Education. „Neural Networks." (17. August 2020), Adresse: https://www.ibm.com/cloud/learn/neural-networks (besucht am 19. September 2021).

[88] N. Donges. „A Guide to RNN: Understanding Recurrent Neural Networks and LSTM Networks," Built In. (29. Juli 2021), Adresse: https://builtin.com/data-science/recurrent-neural-networks-and-lstm (besucht am 21. September 2021).

[89] S. Hochreiter und J. Schmidhuber, „Long Short-Term Memory," *Neural Computation*, Jg. 9, Nr. 8, S. 1735–1780, November 1997. DOI: 10.1162/neco.1997.9.8.1735.

[90] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser und I. Polosukhin, „Attention Is All You Need," 12. Juni 2017. arXiv: 1706.03762 [cs.CL].

[91] H. Kortschak. „Attention and Transformer Models," Medium. (16. November 2020), Adresse: https://towardsdatascience.com/attention-and-transformer-models-fe667f958378 (besucht am 27. September 2021).

[92] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy und S. R. Bowman, „SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems," 2. Mai 2019. arXiv: 1905.00537 [cs.CL].

[93] P. Nayak. „Understanding searches better than ever before," Google LLC. (25. Oktober 2019), Adresse: https://blog.google/products/search/search-language-understanding-bert/ (besucht am 29. September 2021).

[94] M. Z. Mulla. „Cost, Activation, Loss Function, Neural Network, Deep Learning. What are these?" Medium. (26. April 2020), Adresse: https://medium.com/@zeeshanmulla/cost-activation-loss-function-neural-network-deep-learning-what-are-these-91167825a4de (besucht am 22. September 2021).

[95] H. Zulkifli. „Understanding Learning Rates and How It Improves Performance in Deep Learning," Medium. (21. Januar 2018), Adresse: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10 (besucht am 4. Oktober 2021).

[96] J. Delua. „Supervised vs. Unsupervised Learning: What's the Difference?" IBM. (12. Mai 2021), Adresse: https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning (besucht am 23. September 2021).

[97] A. Søgaard, „Semi-Supervised Learning and Domain Adaptation in Natural Language Processing," *Synthesis Lectures on Human Language Technologies*, Jg. 6, Nr. 2, S. 1–103, Mai 2013. DOI: 10.2200/s00497ed1v01y201304hlt021.

[98] R. S. Sutton und A. G. Barto, *Reinforcement Learning: An Introduction*. 2015, ISBN: 978-0262193986.

[99] Tesla. „Tesla AI Day," YouTube. (20. August 2021), Adresse: https://www.youtube.com/watch?v=j0z4FweCy4M (besucht am 23. September 2021).

[100] D. G. Mosquera. „GANs from Scratch 1: A deep introduction. With code in PyTorch and TensorFlow," Medium. (1. Februar 2018), Adresse: https://medium.com/ai-society/gans-from-scratch-1-a-deep-introduction-with-code-in-pytorch-and-tensorflow-cb03cdcdba0f (besucht am 11. Oktober 2021).

[101] A. Karpathy, P. Abbeel, G. Brockman, P. Chen, V. Cheung, R. Duan, I. Goodfellow, D. Kingma, J. Ho, R. Houthooft, T. Salimans, J. Schulman, I. Sutskever und W. Zaremba. „Generative Models," OpenAI. (16. Juni 2016), Adresse: https://openai.com/blog/generative-models/ (besucht am 11. Oktober 2021).

[102] This-Person-Does-not-Exist.com. „Random Face Generator (This Person Does Not Exist)." (2021), Adresse: https://this-person-does-not-exist.com/en (besucht am 12. Oktober 2021).

[103] A. Sagar. „5 Techniques to Prevent Overfitting in Neural Networks," KDnuggets. (5. Dezember 2019), Adresse: https://www.kdnuggets.com/2019/12/5-techniques-prevent-overfitting-neural-networks.html (besucht am 3. Oktober 2021).

[104] DeepAI, Inc. „What is a hyperparameter?" (3. Oktober 2021), Adresse: https://deepai.org/machine-learning-glossary-and-terms/hyperparameter (besucht am 3. Oktober 2021).

[105] A. Radhakrishnan, M. Belkin und C. Uhler, „Overparameterized neural networks implement associative memory," *Proceedings of the National Academy of Sciences*, Jg. 117, Nr. 44, S. 27 162–27 170, Oktober 2020. DOI: 10.1073/pnas.2005013117.

[106] knowledge Transfer. „What is batch size, steps, iteration, and epoch in the neural network?" (14. Dezember 2019), Adresse: https://androidkt.com/batch-size-step-iteration-epoch-neural-network/ (besucht am 3. Oktober 2021).

[107] B. Chen. „Early Stopping in Practice: an example with Keras and TensorFlow 2.0," Medium. (29. Juli 2020), Adresse: https://towardsdatascience.com/a-practical-introduction-to-early-stopping-in-machine-learning-550ac88bc8fd (besucht am 4. Oktober 2021).

[108] D. Vasani. „This thing called Weight Decay," Medium. (29. April 2019), Adresse: https://towardsdatascience.com/this-thing-called-weight-decay-a7cd4bcfccab (besucht am 4. Oktober 2021).

[109] S. Khan. „BERT, RoBERTa, DistilBERT, XLNet — which one to use?“ Medium. (4. September 2019), Adresse: https://towardsdatascience.com/bert-roberta-distilbert-xlnet-which-one-to-use-3d5ab82ba5f8 (besucht am 24. Mai 2021).

[110] J. Devlin, M.-W. Chang, K. Lee und K. Toutanova, „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,“ 11. Oktober 2018. arXiv: 1810.04805 [cs.CL].

[111] Google LLC. „BERT base model (uncased),“ Hugging Face. (18. Mai 2021), Adresse: https://huggingface.co/bert-base-uncased?text=Paris%20is%20the%20[MASK]%20of%20France. (besucht am 8. Oktober 2021).

[112] V. Sanh. „Smaller, faster, cheaper, lighter: Introducing DistilBERT, a distilled version of BERT,“ A Medium Corporation. (28. August 2019), Adresse: https://medium.com/huggingface/distilbert-8cf3380435b5 (besucht am 24. Mai 2021).

[113] V. Sanh, L. Debut, J. Chaumond und T. Wolf, „DistilBERT, a distilled version of BERT smaller, faster, cheaper and lighter,“ 2. Oktober 2019. arXiv: 1910.01108 [cs.CL].

[114] D. Croce, G. Castellucci und R. Basili, „GAN-BERT: Generative Adversarial Learning for Robust Text Classification with a Bunch of Labeled Examples,“ in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Juli 2020, S. 2114–2119. Adresse: https://www.aclweb.org/anthology/2020.acl-main.191.

[115] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer und V. Stoyanov, „RoBERTa: A Robustly Optimized BERT Pretraining Approach,“ 26. Juli 2019. arXiv: 1907.11692 [cs.CL].

[116] R. Scheible, F. Thomczyk, P. Tippmann, V. Jaravine und M. Boeker, *GottBERT: a pure German Language Model*, 2020. arXiv: 2012.02110 [cs.CL].

[117] Facebook Research. „XLM-R: State-of-the-art cross-lingual understanding through self-supervision.“ (11. Juli 2019), Adresse: https://ai.facebook.com/blog/-xlm-r-state-of-the-art-cross-lingual-understanding-through-self-supervision/ (besucht am 18. Oktober 2021).

[118] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer und V. Stoyanov, „Unsupervised Cross-lingual Representation Learning at Scale,“ 5. November 2019. arXiv: 1911.02116 [cs.CL].

[119] Facebook Research. „XLM,“ GitHub. (1. April 2021), Adresse: https://github.com/facebookresearch/XLM (besucht am 18. Oktober 2021).

[120] T. Ray. „Facebook's latest giant language AI hits computing wall at 500 Nvidia GPUs," ZDNet. (12. November 2019), Adresse: https://www.zdnet.com/article/facebooks-latest-giant-language-ai-hits-computing-wall-at-500-nvidia-gpus/ (besucht am 17. September 2021).

[121] P. Shree. „The Journey of Open AI GPT models," Medium. (10. November 2020), Adresse: https://medium.com/walmartglobaltech/the-journey-of-open-ai-gpt-models-32d95b7b7fb2 (besucht am 21. Oktober 2021).

[122] A. Radford, K. Narasimhan, T. Salimans und I. Sutskever, „Improving Language Understandingby Generative Pre-Training," 2018. Adresse: https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf.

[123] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei und I. Sutskever, „Language Models are Unsupervised Multitask Learners," 2019. Adresse: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

[124] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever und D. Amodei, „Language Models are Few-Shot Learners," 28. Mai 2020. arXiv: 2005.14165 [cs.CL].

[125] M. Feiks, *Empirische Sozialforschung mit Python*. Springer Fachmedien Wiesbaden, 2019. DOI: 10.1007/978-3-658-25877-1.

[126] S. Bird, E. Klein und E. Loper, *Natural Language Processing with Python*. O'Reilly UK Ltd., 1. Juli 2009, 504 S., ISBN: 0596516495. Adresse: http://www.nltk.org/book/ (besucht am 24. Mai 2021).

[127] M. R. Aman Kedia, *Hands-On Python Natural Language Processing*. Packt Publishing, 26. Juni 2020, 316 S., ISBN: 1838989595. Adresse: https://www.ebook.de/de/product/39324474/aman_kedia_mayank_rasu_hands_on_python_natural_language_processing.html.

[128] A. Banino, A. P. Badia, J. Walker, T. Scholtes, J. Mitrovic und C. Blundell, „CoBERL: Contrastive BERT for Reinforcement Learning," 12. Juli 2021. arXiv: 2107.05431 [cs.LG].

[129] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy und S. R. Bowman. „SuperGLUE Benchmark Leaderboard Version 2.0." (2. Oktober 2021), Adresse: https://super.gluebenchmark.com/leaderboard/ (besucht am 2. Oktober 2021).

[130] S. Eger, G. G. Şahin, A. Rücklé, J.-U. Lee, C. Schulz, M. Mesgar, K. Swarnkar, E. Simpson und I. Gurevych, „Text Processing Like Humans Do: Visually Attacking and Shielding NLP Systems," 27. März 2019. arXiv: 1903.11508 [cs.CL].

Anhang A: Code Supervised-Learning

```
#!/usr/bin/python
# ========== HYPERPARAMETER ==========
modelname = "bert-base-german-cased"
# distilbert-base-german-cased
# dbmdz/distilbert-base-german-europeana-cased
# bert-base-german-dbmdz-cased
# bert-base-german-dbmdz-uncased
# dbmdz/bert-base-german-europeana-cased
# dbmdz/bert-base-german-europeana-uncased
# uklfr/gottbert-base
# dbmdz/german-gpt2

epochs = 2 # standard = 4
batch_size = 32 # standard = 32
learning_rate = 2e-5 # standard = 2e-5
adam_epsilon = 1e-8 # standard = 1e-8
weight_decay = 0.1 # standard = 0.1
max_len = 128 # maximal = 512
warmup_steps = 0 # default = 0

strip_handles = True # Twitter User Name Handle entfernen
reduceLen = True # Aufeinanderfolgende Buchstaben normalisieren
un_leet = False # Leetspeak entfernen
un_lbr = False # Linebreaks entfernen

save = False # Modell speichern?

from google.colab import drive
drive.mount('/content/drive')

# ========== GPU VORBEREITEN ==========

import tensorflow as tf
device_name = tf.test.gpu_device_name()
import torch

if torch.cuda.is_available():
    device = torch.device("cuda")
    print('%d GPU(s) verfuegbar.' % torch.cuda.device_count())
    print('Verwendetes Modell:', torch.cuda.get_device_name(0))
else:
    print('Keine GPU verfuegbar. Berechnungen laufen auf CPU.')
    device = torch.device("cpu")

!pip install transformers

# ========== DATEN EINLESEN ==========
import pandas as pd
```

```
49
50 # GermEval 2018
51 df_18train = pd.read_csv("/content/drive/MyDrive/MasterThesis/Daten
       /germeval2018.training.txt", sep='\t', names=("comment_text", "
       toxic", "detail"))
52 df_18test = pd.read_csv("/content/drive/MyDrive/MasterThesis/Daten/
       germeval2018.test.txt", sep='\t', names=("comment_text", "toxic"
       , "detail"))
53 del df_18train['detail']
54 del df_18test['detail']
55
56 # GermEval 2019
57 df_19train1 = pd.read_csv("/content/drive/MyDrive/MasterThesis/
       Daten/germeval2019.training_subtask1_2_korrigiert.txt", sep='\t'
       , names=("comment_text", "toxic", "fine"))
58 df_19train2 = pd.read_csv("/content/drive/MyDrive/MasterThesis/
       Daten/germeval2019.training_subtask3.txt", sep='\t', names=("
       comment_text", "toxic", "fine", "expl"))
59 df_19train3 = pd.read_csv("/content/drive/MyDrive/MasterThesis/
       Daten/germeval2019GoldLabelsSubtask1_2.txt", sep='\t', names=("
       comment_text", "toxic", "fine"))
60 df_19train4 = pd.read_csv("/content/drive/MyDrive/MasterThesis/
       Daten/germeval2019GoldLabelsSubtask3.txt", sep='\t', names=("
       comment_text", "toxic", "fine", "expl"))
61 df_19test = pd.read_csv("/content/drive/MyDrive/MasterThesis/Daten/
       germeval2019_Testdata_Subtask3.txt", sep='\t', names=("
       comment_text", "toxic", "fine", "expl"))
62 del df_19train1['fine']
63 del df_19train2['fine']
64 del df_19train2['expl']
65 del df_19train3['fine']
66 del df_19train4['fine']
67 del df_19train4['expl']
68 del df_19test['fine']
69 del df_19test['expl']
70 df_19train = df_19train1.append(df_19train2.append(df_19train3.
       append(df_19train4)))
71
72 # GermEval 2021
73 df_21train = pd.read_csv("/content/drive/MyDrive/MasterThesis/Daten
       /GermEval21_TrainData.csv")
74 df_21test = pd.read_csv("/content/drive/MyDrive/MasterThesis/Daten/
       GermEval21_TestData.csv")
75 del df_21train['comment_id']
76 del df_21train['Sub2_Engaging']
77 del df_21train['Sub3_FactClaiming']
78 df_21train.rename(columns = {'Sub1_Toxic':'toxic'}, inplace = True)
79 del df_21test['comment_id']
80 del df_21test['Sub2_Engaging']
81 del df_21test['Sub3_FactClaiming']
82 df_21test.rename(columns = {'Sub1_Toxic':'toxic'}, inplace = True)
83
```

```
84 # ========== DATEN AUSWAEHLEN ==========
85
86 Trainingsdaten = [df_18train,df_19train,df_21train,df_18test,
       df_19test]
87 Testdaten = [df_21test]
88
89 df_train = pd.DataFrame()
90 df_test = pd.DataFrame()
91 for i in Trainingsdaten:
92   df_train = df_train.append(i, ignore_index=True)
93 for i in Testdaten:
94   df_test = df_test.append(i, ignore_index=True)
95
96 df_train = df_train.replace("OFFENSE", 1)
97 df_train = df_train.replace("OTHER", 0)
98 df_test = df_test.replace("OFFENSE", 1)
99 df_test = df_test.replace("OTHER", 0)
100 print(df_train.shape)
101 print(df_test.shape)
102
103 # ========== TOKENIZER LADEN ==========
104
105 if modelname == "uklfr/gottbert-base": # lade den GottBERT
       tokenizer
106   from transformers import AutoTokenizer, AutoModelForMaskedLM
107   tokenizer = AutoTokenizer.from_pretrained(modelname,strip_handles
       =strip_handles,reduceLen=reduceLen)
108   model = AutoModelForMaskedLM.from_pretrained(modelname)
109 elif modelname == "dbmdz/german-gpt2": # lade den GPT-2 tokenizer
110   from transformers import AutoTokenizer, AutoModelWithLMHead
111   tokenizer = AutoTokenizer.from_pretrained(modelname,strip_handles
       =strip_handles,reduceLen=reduceLen)
112   model = AutoModelWithLMHead.from_pretrained(modelname)
113 else: # lade den BERT Tokenizer
114   from transformers import BertTokenizer
115   tokenizer = BertTokenizer.from_pretrained(modelname,strip_handles
       =strip_handles,reduceLen=reduceLen)
116
117 # ========== SPEZIALTOKEN VERGEBEN ==========
118
119 input_ids = []
120 for sentence in sentences:
121   encoded_sentence = tokenizer.encode(sentence, add_special_tokens
       = True) # Tokenisierung mit CLS, SEP
122   input_ids.append(encoded_sentence) # IDs zuordnen
123
124 from keras.preprocessing.sequence import pad_sequences
125 input_ids = pad_sequences(input_ids, maxlen=max_len, dtype="long",
       value=0, truncating="post", padding="post")
126
127 attention_masks = []
128 for sentence in input_ids:
```

```
129     att_mask = [int(token_id > 0) for token_id in sentence]
130     attention_masks.append(att_mask)
131
132 # ========== TRAIN-VAL-SPLIT ==========
133
134 from sklearn.model_selection import train_test_split
135 # 90% Training - 10% Validierung
136 train_inputs, validation_inputs, train_labels, validation_labels =
        train_test_split(input_ids, labels, random_state=2018, test_size
        =0.1)
137 # Auch fuer die Maskierung
138 train_masks, validation_masks, _, _ = train_test_split(
        attention_masks, labels, random_state=2018, test_size=0.1)
139
140 # ========== KONVERTIERE IN PYTORCH DATENTYPEN ==========
141
142 train_inputs = torch.tensor(train_inputs)
143 validation_inputs = torch.tensor(validation_inputs)
144
145 train_labels = torch.tensor(train_labels)
146 validation_labels = torch.tensor(validation_labels)
147
148 train_masks = torch.tensor(train_masks)
149 validation_masks = torch.tensor(validation_masks)
150
151 from torch.utils.data import TensorDataset, DataLoader,
        RandomSampler, SequentialSampler
152
153 train_data = TensorDataset(train_inputs, train_masks, train_labels)
154 train_sampler = RandomSampler(train_data)
155 train_dataloader = DataLoader(train_data, sampler=train_sampler,
        batch_size=batch_size)
156
157 validation_data = TensorDataset(validation_inputs, validation_masks
        , validation_labels)
158 validation_sampler = SequentialSampler(validation_data)
159 validation_dataloader = DataLoader(validation_data, sampler=
        validation_sampler, batch_size=batch_size)
160
161 # ========== FEINTUNING - KLASSIFIKATOR LADEN ==========
162
163 from transformers import BertForSequenceClassification, AdamW,
        BertConfig
164
165 model = BertForSequenceClassification.from_pretrained(
166     modelname,
167     num_labels = 2, # 2 Label fuer binaere Klassifizierung
168     output_attentions = False,
169     output_hidden_states = False,
170 )
171
172 if torch.cuda.is_available():
```

```
  model.cuda()
else:
  model.cpu()

# ========== LEARNING-RATE-SCHEDULER ==========

optimizer = AdamW(model.parameters(),
    weight_decay = weight_decay,
    lr = learning_rate,
    eps = adam_epsilon
  )

from transformers import get_linear_schedule_with_warmup

total_steps = len(train_dataloader) * epochs
scheduler = get_linear_schedule_with_warmup(optimizer,
                        num_warmup_steps = warmup_steps,
                        num_training_steps = total_steps)

# ========== TRAININGSFORTSCHRITT MESSEN ==========

import numpy as np

# Hilfsfunktion zur Berechnung der Genauigkeit (accuracy)
def flat_accuracy(preds, labels):
    pred_flat = np.argmax(preds, axis=1).flatten()
    labels_flat = labels.flatten()
    return np.sum(pred_flat == labels_flat) / len(labels_flat)

import time
import datetime

def format_time(elapsed):
    # Auf Sekunden runden
    elapsed_rounded = int(round((elapsed)))

    # Formatieren als hh:mm:ss
    return str(datetime.timedelta(seconds=elapsed_rounded))

# ==================== TRAINING ====================

import random

seed_val = 42 # Fester Seed, um Zufaelligkeit wiederholen zu
   koennen

random.seed(seed_val)
np.random.seed(seed_val)
torch.manual_seed(seed_val)
torch.cuda.manual_seed_all(seed_val)

```

```
223 loss_values = [] # Loss-Werte pro Batch runden fuer weitere
    Evaluierung
224
225 for epoch_i in range(0, epochs):
226   # ==================== Training ====================
227   print('======== Epoch {:} / {:} ========'.format(epoch_i + 1,
     epochs))
228   print('Training...')
229
230   t0 = time.time()
231   total_loss = 0 # Loss fuer diesen Batch zuruecksetzen
232
233   model.train()
234   for step, batch in enumerate(train_dataloader):
235     if step % 50 == 0 and not step == 0: # Alle 50 Batches den
     Fortschritt anzeigen.
236         elapsed = format_time(time.time() - t0)
237         print('  Batch {:>5,}  of  {:>5,}.    Elapsed: {:}.'.format
     (step, len(train_dataloader), elapsed))
238
239     b_input_ids = batch[0].to(device)
240     b_input_mask = batch[1].to(device)
241     b_labels = batch[2].to(device)
242
243     model.zero_grad() # Vorher errechnete Gradienten entfernen
244
245     # Daten vorwaerts durch das ANN leiten, um die Labels zu
     generieren
246     outputs = model(b_input_ids, token_type_ids=None,
     attention_mask=b_input_mask, labels=b_labels)
247
248     loss = outputs[0] # Aus den Labels den Loss-Wert fuer den
     aktuellen Batch berechnen
249     total_loss += loss.item() # Lokalen Loss des Batches zu
     globalem Loss des Epochs addieren
250     loss.backward() # Gradienten berechnen durch Backpropagation
251     torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0) #
     Exploding Gradients mit einem Clip von 1 verhindern
252     optimizer.step() # Parameter fuer diesen Schritt anpassen
253     scheduler.step() # Learning-Rate anpassen
254
255   avg_train_loss = total_loss / len(train_dataloader) #
     Durchschnittlichen Loss pro Batch aus gesamten Loss berechnen
256   loss_values.append(avg_train_loss) # Loss des Batches in Liste
     schreiben
257
258   print("\n  Durchschnittlicher Loss: {0:.2f}".format(
     avg_train_loss))
259   print("  Dauer des Trainings fuer diesen Epoch: {:}".format(
     format_time(time.time() - t0)))
260
261   # ==================== VALIDIERUNG ====================
```

```
262
263   print("\n Validierung...")
264
265   t0 = time.time()
266   model.eval()
267
268   # Tracking variables
269   eval_loss, eval_accuracy = 0, 0
270   nb_eval_steps, nb_eval_examples = 0, 0
271
272   for batch in validation_dataloader:
273     batch = tuple(t.to(device) for t in batch) # Batch in GPU
      schreiben
274     b_input_ids, b_input_mask, b_labels = batch # Inputs vom
      DataLoader entpacken
275
276     with torch.no_grad(): # Schnelles Leiten der Daten durch das
      Netz, ohne Gradienten zu berechnen
277         outputs = model(b_input_ids, token_type_ids=None,
      attention_mask=b_input_mask)
278
279     # Berechnetes Label an CPU uebertragen
280     logits = outputs[0].detach().cpu().numpy()
281     label_ids = b_labels.to('cpu').numpy()
282
283     tmp_eval_accuracy = flat_accuracy(logits, label_ids) #
      Genauigkeit (accuracy) fuer diesen Test-Batch berechnen
284     eval_accuracy += tmp_eval_accuracy # lokale Batch-Genauigkeit
      zur globalen Epoch-Genauigkeit addieren
285     nb_eval_steps += 1 # Batchzahl hochzaehlen
286
287   print("  Accuracy: {0:.2f}".format(eval_accuracy/nb_eval_steps))
288   print("  Validationsdauer: {:}\n".format(format_time(time.time()
      - t0)))
289
290 print("Training abgeschlossen!")
291
292 # ========== EVALUIERUNG ANHAND DES TESTDATENSATZES ==========
293
294 sentences = df_test.comment_text.values
295 labels = df_test.toxic.values
296
297 input_ids = []
298
299 for sentence in sentences:
300   encoded_sent = tokenizer.encode(sentence, add_special_tokens =
      True)
301   input_ids.append(encoded_sent)
302
303 input_ids = pad_sequences(input_ids, maxlen=max_len, dtype="long",
      truncating="post", padding="post")
304 attention_masks = []
```

```
305
306 for seq in input_ids:
307   seq_mask = [float(i>0) for i in seq]
308   attention_masks.append(seq_mask)
309
310 prediction_inputs = torch.tensor(input_ids)
311 prediction_masks = torch.tensor(attention_masks)
312 prediction_labels = torch.tensor(labels)
313
314 prediction_data = TensorDataset(prediction_inputs, prediction_masks
    , prediction_labels)
315 prediction_sampler = SequentialSampler(prediction_data)
316 prediction_dataloader = DataLoader(prediction_data, sampler=
    prediction_sampler, batch_size=batch_size)
317
318 # ========== LABEL GENERIEREN ==========
319
320 model.eval()
321
322 predictions , true_labels = [], []
323
324 for batch in prediction_dataloader:
325   batch = tuple(t.to(device) for t in batch)
326   b_input_ids, b_input_mask, b_labels = batch
327
328   with torch.no_grad():
329       outputs = model(b_input_ids, token_type_ids=None,
    attention_mask=b_input_mask)
330
331   logits = outputs[0].detach().cpu().numpy()
332   label_ids = b_labels.to('cpu').numpy()
333
334   predictions.append(logits)
335   true_labels.append(label_ids)
336
337 # ========== ACCURACY ==========
338
339 from sklearn.metrics import accuracy_score
340 import statistics
341 accuracy_batches = []
342 flat_predictions = [item for sublist in predictions for item in
    sublist]
343 flat_predictions = np.argmax(flat_predictions, axis=1).flatten()
344 flat_true_labels = [item for sublist in true_labels for item in
    sublist]
345
346 for i in range(len(true_labels)):
347   accuracy_batches.append(accuracy_score(np.argmax(predictions[i],
    axis=1).flatten(), true_labels[i]))
348
349 print('Toxische Kommentare: %d von %d (%.2f%%)' % (df_test.toxic.
    sum(), len(df_test.toxic), (df_test
```

```
.toxic.sum() / len(df_test.toxic) * 100.0)))
print('Accuracy:', statistics.mean(accuracy_batches))

# ========== PRECISION ==========

from sklearn.metrics import precision_score
print('Precision:', precision_score(flat_true_labels,
    flat_predictions))

# ========== RECALL ==========

from sklearn.metrics import recall_score
print('Recall:', recall_score(flat_true_labels, flat_predictions))

# ========== F1-SCORE ==========

from sklearn.metrics import f1_score
print('F1:', f1_score(flat_true_labels, flat_predictions, average='
    macro'))

# ========== MCC-SCORE ==========

from sklearn.metrics import matthews_corrcoef
matthews_set = []
for i in range(len(true_labels)):
  pred_labels_i = np.argmax(predictions[i], axis=1).flatten()
  matthews_set.append(matthews_corrcoef(true_labels[i],
    pred_labels_i))
print('MCC:', matthews_corrcoef(flat_true_labels, flat_predictions)
    )

# ========== MODELL SPEICHERN ==========

if (save):
  import os
  output_dir = '/content/drive/MyDrive/MasterThesis/Finetuning'
  if not os.path.exists(output_dir):
    os.makedirs(output_dir)

  model_to_save = model.module if hasattr(model, 'module') else
    model # Take care of distributed/parallel training
  model_to_save.save_pretrained(output_dir)
  tokenizer.save_pretrained(output_dir)

  torch.save(args, os.path.join(output_dir, 'training_args.bin')) #
    Hyperparameter schreiben
```

Anhang B: Code Semi-Supervised-Learning

```
#!/usr/bin/python
# ========== HYPERPARAMETER ==========
modelname = "bert-base-german-cased"
# bert-base-german-dbmdz-cased
# bert-base-german-dbmdz-uncased
# dbmdz/bert-base-german-europeana-cased
# dbmdz/bert-base-german-europeana-uncased

num_train_epochs = 1 # standard = 1
Learning_Rate = 5e-5 # standard = 5e-5
batch_size = 32 # standard = 32
learning_rate_discriminator = Learning_Rate # standard = 5e-5
learning_rate_generator = Learning_Rate # standard = 5e-5
epsilon = 1e-8 # standard = 1e-8
weight_decay = 0.1 # standard = 0.1
max_seq_length = 64 # maximal = 512
multi_gpu = True # standard = True
apply_scheduler = True # standard = False
frac = 0.5 # Anteil annotierter Daten
warmup_proportion = frac/10 # standard = 0.1
print_each_n_step = 40

num_hidden_layers_g = 1; # standard = 1 Hidden Layer Generator
num_hidden_layers_d = 1; # standard = 1 Hidden Layer Discriminator
noise_size = 100  # standard = 100 Zufallsvektoren
out_dropout_rate = 0.2 # standard = 0.2 zu Discriminator Input
    Vektoren

apply_balance = True
strip_handles = False # Twitter User Name Handle entfernen
reduceLen = False # Aufeinanderfolgende Buchstaben normalisieren
un_leet = False
un_lbr = False

save = False # Modell speichern?

# ========== SETUP ==========

from google.colab import drive
drive.mount('/content/drive')

!pip install transformers==4.3.2
import pandas as pd
import torch
import io
import torch.nn.functional as F
import random
import numpy as np
```

```
import time
import math
import datetime
import torch.nn as nn
from transformers import *
from torch.utils.data import TensorDataset, DataLoader,
    RandomSampler, SequentialSampler
!pip install torch==1.7.1+cu101 torchvision==0.8.2+cu101 -f https
    ://download.pytorch.org/whl/torch_stable.html
!pip install sentencepiece

seed_val = 42 # Zufallszahlen mit Seed versehen
random.seed(seed_val)
np.random.seed(seed_val)
torch.manual_seed(seed_val)
if torch.cuda.is_available():
  torch.cuda.manual_seed_all(seed_val)

# ========== GPU VORBEREITEN ==========

if torch.cuda.is_available():
    device = torch.device("cuda")
    print('%d GPU(s) verfuegbar.' % torch.cuda.device_count())
    print('Verwendetes Modell:', torch.cuda.get_device_name(0))
else:
    print('Keine GPU verfuegbar. Berechnungen laufen auf CPU.')
    device = torch.device("cpu")

!git clone https://github.com/crux82/ganbert

# ========== DATEN EINLESEN ==========
import pandas as pd

# GermEval 2018
df_18train = pd.read_csv("/content/drive/MyDrive/MasterThesis/Daten
    /germeval2018.training.txt", sep='\t', names=("comment_text", "
    toxic", "detail"))
df_18test = pd.read_csv("/content/drive/MyDrive/MasterThesis/Daten/
    germeval2018.test.txt", sep='\t', names=("comment_text", "toxic"
    , "detail"))
del df_18train['detail']
del df_18test['detail']

# GermEval 2019
df_19train1 = pd.read_csv("/content/drive/MyDrive/MasterThesis/
    Daten/germeval2019.training_subtask1_2_korrigiert.txt", sep='\t'
    , names=("comment_text", "toxic", "fine"))
df_19train2 = pd.read_csv("/content/drive/MyDrive/MasterThesis/
    Daten/germeval2019.training_subtask3.txt", sep='\t', names=("
    comment_text", "toxic", "fine", "expl"))
```

```
88 df_19train3 = pd.read_csv("/content/drive/MyDrive/MasterThesis/
       Daten/germeval2019GoldLabelsSubtask1_2.txt", sep='\t', names=("
       comment_text", "toxic", "fine"))
89 df_19train4 = pd.read_csv("/content/drive/MyDrive/MasterThesis/
       Daten/germeval2019GoldLabelsSubtask3.txt", sep='\t', names=("
       comment_text", "toxic", "fine", "expl"))
90 df_19test = pd.read_csv("/content/drive/MyDrive/MasterThesis/Daten/
       germeval2019_Testdata_Subtask3.txt", sep='\t', names=("
       comment_text", "toxic", "fine", "expl"))
91 del df_19train1['fine']
92 del df_19train2['fine']
93 del df_19train2['expl']
94 del df_19train3['fine']
95 del df_19train4['fine']
96 del df_19train4['expl']
97 del df_19test['fine']
98 del df_19test['expl']
99 df_19train = df_19train1.append(df_19train2.append(df_19train3.
       append(df_19train4)))
100
101 # GermEval 2021
102 df_21train = pd.read_csv("/content/drive/MyDrive/MasterThesis/Daten
       /GermEval21_TrainData.csv")
103 df_21test = pd.read_csv("/content/drive/MyDrive/MasterThesis/Daten/
       GermEval21_TestData.csv")
104 del df_21train['comment_id']
105 del df_21train['Sub2_Engaging']
106 del df_21train['Sub3_FactClaiming']
107 df_21train.rename(columns = {'Sub1_Toxic':'toxic'}, inplace = True)
108 del df_21test['comment_id']
109 del df_21test['Sub2_Engaging']
110 del df_21test['Sub3_FactClaiming']
111 df_21test.rename(columns = {'Sub1_Toxic':'toxic'}, inplace = True)
112
113 # ========== DATEN AUSWAEHLEN ==========
114
115 Trainingsdaten = [df_18train,df_19train,df_21train,df_18test,
       df_19test]
116 Testdaten = [df_21test]
117
118 df_train = pd.DataFrame()
119 df_test = pd.DataFrame()
120 for i in Trainingsdaten:
121   df_train = df_train.append(i, ignore_index=True)
122 for i in Testdaten:
123   df_test = df_test.append(i, ignore_index=True)
124
125 df_train = df_train.replace("OFFENSE", 1)
126 df_train = df_train.replace("OTHER", 0)
127 df_test = df_test.replace("OFFENSE", 1)
128 df_test = df_test.replace("OTHER", 0)
129 print(df_train.shape)
```

```
print(df_test.shape)

label_list = [0, 1, 2]

df_train_for_ganbert = df_train.sample(frac=frac) # Anteil Label entfernen
df_unlabeled = df_train.drop(df_train_for_ganbert.index)

# Daten ohne Label zu UNK (unknown)
for i in df_unlabeled.index:
    df_unlabeled.at[i, "toxic"] = 2

def get_examples(df):
    examples = []
    for index, row in df.iterrows():
        examples.append((row['comment_text'], row['toxic']))
    return examples

labeled_examples = get_examples(df_train_for_ganbert)
unlabeled_examples = get_examples(df_unlabeled)
test_examples = get_examples(df_test)

transformer = AutoModel.from_pretrained(modelname)
tokenizer = AutoTokenizer.from_pretrained(modelname,strip_handles=strip_handles,reduceLen=reduceLen)

# ========== DATA LOADER ==========

def generate_data_loader(input_examples, label_masks, label_map, do_shuffle = False, balance_label_examples = False):
  # Anteil der Label ermitteln
  examples = []
  num_labeled_examples = 0
  for label_mask in label_masks:
    if label_mask:
      num_labeled_examples += 1
  label_mask_rate = num_labeled_examples/len(input_examples)

  # Wenn noetig, gleich die Datenklassen aus
  for index, ex in enumerate(input_examples):
    if label_mask_rate == 1 or not balance_label_examples:
      examples.append((ex, label_masks[index]))
    else:
      if label_masks[index]:
        balance = int(1/label_mask_rate)
        balance = int(math.log(balance,2))
        if balance < 1:
          balance = 1
        for b in range(0, int(balance)):
          examples.append((ex, label_masks[index]))
      else:
        examples.append((ex, label_masks[index]))
```

```
179
180   input_ids = []
181   input_mask_array = []
182   label_mask_array = []
183   label_id_array = []
184
185   # Tokenisierung
186   for (text, label_mask) in examples:
187     encoded_sent = tokenizer.encode(text[0], add_special_tokens=
      True, max_length=max_seq_length, padding="max_length",
      truncation=True)
188     input_ids.append(encoded_sent)
189     label_id_array.append(label_map[text[1]])
190     label_mask_array.append(label_mask)
191
192   for sent in input_ids:
193     att_mask = [int(token_id > 0) for token_id in sent]
194     input_mask_array.append(att_mask)
195
196   # In Tensoren konvertieren
197   input_ids = torch.tensor(input_ids)
198   input_mask_array = torch.tensor(input_mask_array)
199   label_id_array = torch.tensor(label_id_array, dtype=torch.long)
200   label_mask_array = torch.tensor(label_mask_array)
201   dataset = TensorDataset(input_ids, input_mask_array,
      label_id_array, label_mask_array)
202
203   if do_shuffle:
204     sampler = RandomSampler
205   else:
206     sampler = SequentialSampler
207
208   return DataLoader(
209               dataset,
210               sampler = sampler(dataset),
211               batch_size = batch_size,
212               drop_last=True)
213
214 def format_time(elapsed):
215     elapsed_rounded = int(round((elapsed)))
216     return str(datetime.timedelta(seconds=elapsed_rounded))
217
218 # ========== LABEL VERWALTEN ==========
219
220 label_map = {}
221
222 for (i, label) in enumerate(label_list):
223   label_map[label] = i
224
225 train_examples = labeled_examples
226 train_label_masks = np.ones(len(labeled_examples), dtype=bool)
227
```

```
228 if unlabeled_examples:
229   train_examples = train_examples + unlabeled_examples
230   tmp_masks = np.zeros(len(unlabeled_examples), dtype=bool)
231   train_label_masks = np.concatenate([train_label_masks,tmp_masks])
232 
233 train_dataloader = generate_data_loader(train_examples,
      train_label_masks, label_map, do_shuffle = True,
      balance_label_examples = apply_balance)
234 test_label_masks = np.ones(len(test_examples), dtype=bool)
235 test_dataloader = generate_data_loader(test_examples,
      test_label_masks, label_map, do_shuffle = False,
      balance_label_examples = False)
236 
237 # ========== GENERATOR DEFINIEREN ==========
238 
239 class Generator(nn.Module):
240     def __init__(self, noise_size=100, output_size=512,
      hidden_sizes=[512], dropout_rate=0.1):
241         super(Generator, self).__init__()
242         layers = []
243         hidden_sizes = [noise_size] + hidden_sizes
244         for i in range(len(hidden_sizes)-1):
245             layers.extend([nn.Linear(hidden_sizes[i], hidden_sizes[
      i+1]), nn.LeakyReLU(0.2, inplace=True), nn.Dropout(dropout_rate)
      ])
246 
247         layers.append(nn.Linear(hidden_sizes[-1],output_size))
248         self.layers = nn.Sequential(*layers)
249 
250     def forward(self, noise):
251         output_rep = self.layers(noise)
252         return output_rep
253 
254 # ========== DISCRIMINATOR DEFINIEREN ==========
255 
256 class Discriminator(nn.Module):
257     def __init__(self, input_size=512, hidden_sizes=[512],
      num_labels=2, dropout_rate=0.1):
258         super(Discriminator, self).__init__()
259         self.input_dropout = nn.Dropout(p=dropout_rate)
260         layers = []
261         hidden_sizes = [input_size] + hidden_sizes
262         for i in range(len(hidden_sizes)-1):
263             layers.extend([nn.Linear(hidden_sizes[i], hidden_sizes[
      i+1]), nn.LeakyReLU(0.2, inplace=True), nn.Dropout(dropout_rate)
      ])
264 
265         self.layers = nn.Sequential(*layers)
266         self.logit = nn.Linear(hidden_sizes[-1],num_labels+1)
267         self.softmax = nn.Softmax(dim=-1)
268 
269     def forward(self, input_rep):
```

```
270         input_rep = self.input_dropout(input_rep)
271         last_rep = self.layers(input_rep)
272         logits = self.logit(last_rep)
273         probs = self.softmax(logits)
274         return last_rep, logits, probs
275 
276 # ========== MODELL IN GPU LADEN ==========
277 
278 config = AutoConfig.from_pretrained(modelname)
279 hidden_size = int(config.hidden_size)
280 hidden_levels_g = [hidden_size for i in range(0,
    num_hidden_layers_g)]
281 hidden_levels_d = [hidden_size for i in range(0,
    num_hidden_layers_d)]
282 
283 generator = Generator(noise_size=noise_size, output_size=
    hidden_size, hidden_sizes=hidden_levels_g, dropout_rate=
    out_dropout_rate)
284 discriminator = Discriminator(input_size=hidden_size, hidden_sizes=
    hidden_levels_d, num_labels=len(label_list), dropout_rate=
    out_dropout_rate)
285 
286 if torch.cuda.is_available():
287   generator.cuda()
288   discriminator.cuda()
289   transformer.cuda()
290   if multi_gpu:
291     transformer = torch.nn.DataParallel(transformer)
292 
293 # ========== FEINTUNING ==========
294 
295 training_stats = []
296 total_t0 = time.time()
297 
298 transformer_vars = [i for i in transformer.parameters()]
299 d_vars = transformer_vars + [v for v in discriminator.parameters()]
300 g_vars = [v for v in generator.parameters()]
301 
302 dis_optimizer = torch.optim.AdamW(d_vars, lr=
    learning_rate_discriminator)
303 gen_optimizer = torch.optim.AdamW(g_vars, lr=
    learning_rate_generator)
304 
305 if apply_scheduler:
306   num_train_examples = len(train_examples)
307   num_train_steps = int(num_train_examples / batch_size *
     num_train_epochs)
308   num_warmup_steps = int(num_train_steps * warmup_proportion)
309 
310   scheduler_d = get_constant_schedule_with_warmup(dis_optimizer,
311                                 num_warmup_steps = num_warmup_steps)
312   scheduler_g = get_constant_schedule_with_warmup(gen_optimizer,
```

```
                                num_warmup_steps = num_warmup_steps)

for epoch_i in range(0, num_train_epochs):
    print("")
    print('======== Epoch {:} / {:} ========'.format(epoch_i + 1,
    num_train_epochs))
    print('Training...')

    t0 = time.time()

    tr_g_loss = 0
    tr_d_loss = 0

    transformer.train()
    generator.train()
    discriminator.train()

    for step, batch in enumerate(train_dataloader):

        if step % print_each_n_step == 0 and not step == 0:
            elapsed = format_time(time.time() - t0)
            print('  Batch {:>5,}  of  {:>5,}.    Dauer: {:}.'.
    format(step, len(train_dataloader), elapsed))

        b_input_ids = batch[0].to(device)
        b_input_mask = batch[1].to(device)
        b_labels = batch[2].to(device)
        b_label_mask = batch[3].to(device)

        model_outputs = transformer(b_input_ids, attention_mask=
    b_input_mask)
        hidden_states = model_outputs[-1]

        noise = torch.zeros(b_input_ids.shape[0],noise_size, device
    =device).uniform_(0, 1)
        gen_rep = generator(noise)

        disciminator_input = torch.cat([hidden_states, gen_rep],
    dim=0)
        features, logits, probs = discriminator(disciminator_input)

        features_list = torch.split(features, batch_size)
        D_real_features = features_list[0]
        D_fake_features = features_list[1]

        logits_list = torch.split(logits, batch_size)
        D_real_logits = logits_list[0]
        D_fake_logits = logits_list[1]

        probs_list = torch.split(probs, batch_size)
        D_real_probs = probs_list[0]
        D_fake_probs = probs_list[1]
```

```
360
361         g_loss_d = -1 * torch.mean(torch.log(1 - D_fake_probs[:,-1]
     + epsilon))
362         g_feat_reg = torch.mean(torch.pow(torch.mean(
    D_real_features, dim=0) - torch.mean(D_fake_features, dim=0), 2)
    )
363         g_loss = g_loss_d + g_feat_reg
364
365         logits = D_real_logits[:,0:-1]
366         log_probs = F.log_softmax(logits, dim=-1)
367
368         label2one_hot = torch.nn.functional.one_hot(b_labels, len(
    label_list))
369         per_example_loss = -torch.sum(label2one_hot * log_probs,
    dim=-1)
370         per_example_loss = torch.masked_select(per_example_loss,
    b_label_mask.to(device))
371         labeled_example_count = per_example_loss.type(torch.float32
    ).numel()
372
373         if labeled_example_count == 0:
374           D_L_Supervised = 0
375         else:
376           D_L_Supervised = torch.div(torch.sum(per_example_loss.to(
    device)), labeled_example_count)
377
378         D_L_unsupervised1U = -1 * torch.mean(torch.log(1 -
    D_real_probs[:, -1] + epsilon))
379         D_L_unsupervised2U = -1 * torch.mean(torch.log(D_fake_probs
    [:, -1] + epsilon))
380         d_loss = D_L_Supervised + D_L_unsupervised1U +
    D_L_unsupervised2U
381
382         gen_optimizer.zero_grad()
383         dis_optimizer.zero_grad()
384
385         g_loss.backward(retain_graph=True)
386         d_loss.backward()
387
388         gen_optimizer.step()
389         dis_optimizer.step()
390
391         tr_g_loss += g_loss.item()
392         tr_d_loss += d_loss.item()
393
394         if apply_scheduler:
395           scheduler_d.step()
396           scheduler_g.step()
397
398     avg_train_loss_g = tr_g_loss / len(train_dataloader)
399     avg_train_loss_d = tr_d_loss / len(train_dataloader)
400
```

```
401     training_time = format_time(time.time() - t0)
402 
403     print("")
404     print("  Loss Generator: {0:.3f}".format(avg_train_loss_g))
405     print("  Loss Discriminator: {0:.3f}".format(avg_train_loss_d))
406     print("  Epoch-Trainingsdauer: {:}".format(training_time))
407 
408     print("Modelle werden gespeichert...")
409     torch.save(transformer, 'transformer')
410     torch.save(discriminator, 'discriminator')
411 
412 # ========== VALIDIERUNG ==========
413 
414     print("")
415     print("Running Test...")
416 
417     t0 = time.time()
418 
419     transformer.eval()
420     discriminator.eval()
421     generator.eval()
422 
423     total_test_accuracy = 0
424 
425     total_test_loss = 0
426     nb_test_steps = 0
427 
428     all_preds = []
429     all_labels_ids = []
430 
431     nll_loss = torch.nn.CrossEntropyLoss(ignore_index=-1)
432 
433     for batch in test_dataloader:
434 
435         b_input_ids = batch[0].to(device)
436         b_input_mask = batch[1].to(device)
437         b_labels = batch[2].to(device)
438 
439         with torch.no_grad():
440             model_outputs = transformer(b_input_ids, attention_mask
    =b_input_mask)
441             hidden_states = model_outputs[-1]
442             _, logits, probs = discriminator(hidden_states)
443             filtered_logits = logits[:,0:-1]
444             total_test_loss += nll_loss(filtered_logits, b_labels)
445         _, preds = torch.max(filtered_logits, 1)
446         all_preds += preds.detach().cpu()
447         all_labels_ids += b_labels.detach().cpu()
448 
449     all_preds = torch.stack(all_preds).numpy()
450     all_labels_ids = torch.stack(all_labels_ids).numpy()
```

```
451     test_accuracy = np.sum(all_preds == all_labels_ids) / len(
        all_preds)
452     print("  Accuracy: {0:.3f}".format(test_accuracy))
453 
454     avg_test_loss = total_test_loss / len(test_dataloader)
455     avg_test_loss = avg_test_loss.item()
456 
457     test_time = format_time(time.time() - t0)
458 
459     print("  Test Loss: {0:.3f}".format(avg_test_loss))
460     print("  Test took: {:}".format(test_time))
461 
462     training_stats.append(
463         {
464             'epoch': epoch_i + 1,
465             'Training Loss generator': avg_train_loss_g,
466             'Training Loss discriminator': avg_train_loss_d,
467             'Valid. Loss': avg_test_loss,
468             'Valid. Accur.': test_accuracy,
469             'Training Time': training_time,
470             'Test Time': test_time
471         }
472     )
473 
474 # ========== EVALUIERUNG ==========
475 
476 transformer = torch.load('transformer')
477 discriminator = torch.load('discriminator')
478 
479 from sklearn.metrics import accuracy_score
480 from sklearn.metrics import precision_score
481 from sklearn.metrics import recall_score
482 from sklearn.metrics import f1_score
483 from sklearn.metrics import matthews_corrcoef
484 print('\n==== Genauigkeitsmetriken ====')
485 print('Accuracy: %.2f' % np.multiply(100, accuracy_score(
        all_labels_ids, all_preds)), '%')
486 print('Precision: %.2f' % np.multiply(100, precision_score(
        all_labels_ids, all_preds)), '%')
487 print('Recall: %.2f' % np.multiply(100, recall_score(all_labels_ids
        , all_preds)), '%')
488 print('F1: %.2f' % np.multiply(100, f1_score(all_labels_ids,
        all_preds, average='macro')), '%')
489 print('MCC: %.2f' % np.multiply(100, matthews_corrcoef(
        all_labels_ids, all_preds)), '%')
```

Anhang C: Code Jigsaw Genauigkeitsmetriken

```
#!/usr/bin/python
df = pd.read_csv("/content/drive/MyDrive/MasterThesis/Daten/
    GermEval Perspective.csv")
pred = np.round(df['Toxizitaet'].values)
true_labels = df['Angreifend'].values

for i, j in enumerate(true_labels):
  if j == "OTHER":
    true_labels[i] = int(0)
  else:
    true_labels[i] = int(1)

pred = np.array(pred, dtype='int')[-3398:]
true_labels = np.array(true_labels, dtype='int')[-3398:]

print('Accuracy: %.2f' % np.multiply(100, accuracy_score(
    true_labels, pred)), '%')
print('Precision: %.2f' % np.multiply(100, precision_score(
    true_labels, pred)), '%')
print('Recall: %.2f' % np.multiply(100, recall_score(true_labels,
    pred)), '%')
print('F1: %.2f' % np.multiply(100, f1_score(true_labels, pred,
    average='macro')), '%')
print('MCC: %.2f' % np.multiply(100, matthews_corrcoef(true_labels,
     pred)), '%')
```

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 01.12.2021