FernUniversität in Hagen
Fakultät für Mathematik und Informatik

Artificial Intelligence Group

Erkennung von Hate Speech mit Twitter

Seminararbeit

vorgelegt von

Elena Marion Friedrich, Nasiba Tuychieva, Sven Ole Wall, Imran Nteli Chalil, Christian Engels

Zusammenfassung

Soziale Medienplattformen haben die digitale Kommunikation erheblich verändert, bringen jedoch auch Herausforderungen mit sich, insbesondere im Zusammenhang mit Hate Speech. Studien zeigen, dass ein großer Anteil junger Erwachsener online mit hasserfüllten Inhalten konfrontiert wird und diese negative psychologische sowie gesellschaftliche Auswirkungen haben können. Um dieser Problematik entgegenzuwirken, sind automatische Erkennungsverfahren zur Identifikation von Hate Speech erforderlich. Vor diesem Hintergrund ist es das Ziel dieser Arbeit zu untersuchen, inwiefern sich verschiedene Machine- und Deep-Learning-Ansätze zur automatischen Erkennung von Hate Speech eignen.

1 Einleitung

Soziale Medienplattformen wie Twitter haben die digitale Kommunikation und Interaktion revolutioniert, beinhalten dabei jedoch häufig auch Aspekte, welche sich negativ auf ihre Nutzer auswirken [4]. So hat eine Umfrage gezeigt, dass 42% bis 67% der jungen Erwachsenen hassvolle und degradierende Inhalte beobachteten [39, 32] und 21% selbst Opfer von Hassrede (engl.„Hate Speech“) im Internet wurden [10]. Dass dies problematisch ist, wird mit Blick auf die vielfältigen negativen Folgen von Hate Speech, wie Angst und Depression bei den betroffenen Personen [13], deutlich. Darüber hinaus deuten Forschungsergebnisse darauf hin, dass Hate Speech im Internet Ausgrenzung und physische Gewalt bedingen kann. So konnte basierend auf den Ergebnissen von [46] häufiger Kontakt mit Hate Speech gegen bestimmte Gruppen mit verstärkten Vorurteilen gegenüber diesen in Verbindung gebracht werden und [21] zeigen, dass flüchtlingsfeindliche Rhetorik auf Facebook zu höheren Kriminalitätsraten gegen Flüchtlinge führen kann. Diese Befunde verdeutlichten die Wichtigkeit, Hate Speech auf sozialen Medienplattformen zu erkennen und entgegenzuwirken.

Definition Hate Speech: Für den weiteren Verlauf unseres Berichts ist es wichtig festzuhalten, dass im Rahmen der Projektarbeit Hate Speech angelehnt an das Cambridge Dictionary [17] als menschenverachtende Aussagen, die Einzelpersonen oder Gruppen aufgrund von Merkmalen wie Herkunft, Geschlecht oder Religion abwerten definiert wird. Zudem berücksichtigen wir gezielte Äußerungen, die mit Drohungen oder Aufrufen zur Gewalt verbunden sind als Hate Speech.

2 Aufgabenverteilung

Neben Arbeiten, an denen jeder beteiligt war wie z.B. Data Exploration, Ausarbeitung oder Testdaten-Labeling kümmerten sich die einzelnen Teammitglieder speziell um folgende Themen: **Imran Nteli Chalil** wirkte bei der Datenvorverarbeitung mit, verfasste die allgemeinen Kapitel der Ausarbeitung und recherchierte zu ähnlichen Arbeiten sowie Evaluationsmetriken der Hate-Speech-Erkennung. Im Experiment arbeitete er mit Deep Learning, insbesondere GRU und BERT. **Elena Marion Friedrich** evaluierte Resampling-Ansätze, erstellte die Präsentationen und übernahm die allgemeine Team-Koordination sowie zahlreiche Review- und Schreibaufgaben im Rahmen der Erstellung des Abschlussberichtes. Im Rahmen des Experiments beschäftigte sie sich mit der Support Vector Machine. **Nasiba Tuychieva** wirkte insbesondere bei den Aktivitäten in Datenbereinigung bzw. Datenvorbereitung und Entwicklung der Vektorisierungsmethoden mit. Im Rahmen des Experiments beschäftigte sie sich mit der RNN LSTM. Wirkte ebenfalls bei der Erstellung der Präsentationen unterstützend. **Sven Ole Wall** kümmerte sich um die Recherche zur Vektorisierung und der Entwicklung und zentralen Bereitstellung der Vektorisierungsfunktionen. Im Experiment befasste er sich mit dem Naive Bayes-Klassifikator. **Christian Engels** hat bei der Datenbereinigung, -Vorbereitung und -Erweiterung sowie der Gesamtevaluation mitgewirkt. Im Experiment war er für Ensemble- sowie RoBERTa-Modelle zuständig und hat für die Abschlusspräsentation die Web-Applikation erstellt.

3 Teaminterne Organisation

Das Team koordinierte sich durch regelmäßige Meetings in Zoom und digitale Tools. Zu Projektbeginn wurden Aufgaben wie die explorative Analyse, Problemidentifikation und Datenvorverarbeitung parallel bearbeitet und in den Meetings zusammengeführt, was ein tiefes Verständnis und vielfältige Lösungsansätze förderte. Nach der Datenbereinigung und Zieldefinition erfolgte eine auf Stärken basierte Aufgabenverteilung zur effizienten Parallelisierung. GitHub wurde sowohl als zentrales Code-Repository als auch als Projektmanagement-Tool genutzt. Neben der Versionskontrolle ermöglichte es die strukturierte Aufgabenverwaltung über Projekte, Boards und Issues. Aufgaben konnten klar zugewiesen, der Fortschritt nachverfolgt und Änderungen im Code transparent dokumentiert werden. WhatsApp diente als primäres Kommunikationstool für spontane Absprachen. Für die technische Umsetzung

wurde Python mit Jupyter Notebooks gewählt – aufgrund geeigneter Bibliotheken und vorhandener Expertise. Die Notebooks ermöglichten eine enge Verknüpfung von Code, Visualisierungen und Dokumentation. Wiederverwendbare Funktionen, z.B. zur Vektorisierung, sorgten für konsistente Datenverarbeitung. Herausfordernd waren die langen Trainingszeiten der Modelle, die lokal nur eingeschränkt möglich waren. Durch den Wechsel auf Cloud-Ressourcen konnten diese Engpässe jedoch kompensiert werden.

4 Datensätze und Problemstellung

Wie bereits in Kapitel 1 dargelegt, besteht ein hoher Bedarf an automatisierten Methoden zur Erkennung von Hate Speech in sozialen Medien. Die Herausforderung liegt in der subtilen, mehrdeutigen und kontextabhängigen Formulierung solcher Inhalte [48]. Zudem erschwert die Unterrepräsentation von Hasskommentaren in vielen Datensätzen das Modelltraining und kann zu Verzerrungen führen. Die Kombination aus sprachlicher Variabilität, unausgeglichenen Daten und der Notwendigkeit hoher Klassifikationsgenauigkeit erfordert den Einsatz geeigneter Methoden zur Datenaufbereitung für ein optimales Modelltraining.

4.1 Initialer Datensatz

Der durch Analytics Vidhya bereitgestellte Datensatz wurde zuletzt 2019 aktualisiert [7] und enthält eine Spalte für Tweets sowie deren binäre Hate-Speech-Labels [40]. Da der Datensatz im Rahmen eines Wettbewerbs veröffentlicht wurde, und es hierbei üblich ist, dass die Ergebnisse eingereicht und anschließend durch den Veranstalter ausgewertet werden, ist der Testdatensatz nicht gelabelt. Aus diesem Grund kann dieser Datensatz nicht für eine maschinelle Evaluation durch das Team verwendet werden. Zudem fielen bei der explorativen Datenanalyse Fälle auf, in denen Tweets offensichtlich falsch gekennzeichnet waren (bspw. *"retweet if you agree!"* und *"aloha and peace symbol on we oppose fascism and all warsforoil"* galt als Hate Speech). Darüber hinaus war der Datensatz mit einem Verhältnis von über 1:10 stark unausgewogen. Aufgrund der mangelhaften Datenqualität und geringen Testdatenmenge wurden Over-/Undersampling verworfen. Angesichts dieser Schwierigkeiten entschied sich das Team, nach weiteren Datensätzen zu suchen, die eine höhere Qualität und ein besseres Klassengleichgewicht aufweisen.

4.2 Erweiterung des Datensatzes

Als Akzeptanzkriterien für die neu hinzuzufügenden Datensätze wurde festgelegt, dass diese ausschließlich Twitter-Daten enthalten und bereits in anderen Forschungsarbeiten zur Erkennung von Hassrede verwendet worden sein sollten. Insgesamt konnten so drei geeignete Datensätze identifiziert werden [22, 45, 14]. Strukturell besitzen diese Datensätze eine sehr große Ähnlichkeit zum Ausgangsdatensatz. Kombiniert

mit den Ausgangsdaten ergibt sich somit ein Gesamtdatensatz mit ca. 207k Einträgen und einem Verhältnis (ja/nein) von 1:2.81.

4.3 Vorverarbeitung

Angelehnt an best practices ([8, 23]) wurden zur Vorbereitung der Daten umfangreiche Vorverarbeitungsschritte durchgeführt, darunter die Normalisierung (z.B. Kleinschreibung, Tokenisierung, Entfernung von URLs, Sonderzeichen, Emojis, Zahlen und häufigen irrelevanten Wörtern) sowie die semantische Aufbereitung durch Lemmatisierung und das Entfernen von Stoppwörtern. Dabei wurde besonderer Wert auf die **logische Reihenfolge** der Schritte gelegt. So erfolgte beispielsweise erst das Auftrennen zusammengeschriebener Wörter, bevor das Stemming durchgeführt wurde, um so einen bestmöglichen Effekt der einzelnen Schritte zu gewährleisten. Beim Testen mit unterschiedlichen **Stemming / Lemmatization** Libraries hat sich gezeigt, dass viele Libraries nur ihren eigenen Zweck gut erfüllen, wordurch die Reihenfolge der Anwendung relevant wird. Der PorterStemmer verändert z.B. „writing“ korrekt zu „write“ , jedoch auch „morning“ zu „morn“ . Der WordNet Lemmatisierer in der Grundform belässt „morning“, verändert jedoch auch „writing“ nicht. Dies kann mithilfe von Part-Of-Speech-Tagging umgangen werden, wodurch Nomen, Verben usw. erkannt und besser verarbeitet werden können. Mithilfe der Library spaCy können so beide Verarbeitungsschritte gleichzeitig durchgeführt werden und die bestmöglichen Ergebnisse erzielt werden [18], weshalb sich das Team für die Verwendung dieser Library entschied. Die Auflösung von **Negationen**, wie „not good“ zu „bad“, wurde aufgrund fehlender umfassender Libraries und der aufwändigen Implementierung mit wortbasierten Ansätzen nicht berücksichtigt. Ebenso wurde keine **Rechtschreibkorrektur** durchgeführt. Zwar gibt es hierfür Libraries wie TextBlob, die falsch geschriebene Wörter korrigieren können, jedoch zeigte sich in Tests, dass diese Library rechenintensiv ist und nicht fehlerfrei korrigiert. Für das Training von Deep-Learning-Modellen wurden bestimmte Vorverarbeitungsschritte anders gehandhabt oder ausgelassen, da eine zu starke Vorverarbeitung sich hier negativ auf die Modellleistung auswirken kann [20].

Um den Datensatz für die ML- und Deep Learning-Ansätze verwendbar zu machen muss der Text durch **Vektorisierung** in eine numerische Form gebracht werden. Nach einer Recherche anhand einschlägiger Forschungsarbeiten [3, 11, 2, 35, 27] sowie eines Tests wurden die folgenden drei Ansätze für das weitere Arbeiten berücksichtigt: TF-IDF [37], Word2Vec [51] und GloVe [31]. Um die semantischen Stärken von GloVe optimal zu nutzen, wurde sich für ein 200-dimensionales Modell entschieden das speziell auf Tweets trainiert wurde. Um den speziellen Anforderungen der DL-Modelle gerecht zu werden, wurden bei diesen auch eine Vektorisierung mittels vortrainierter Embeddings innerhalb des Embedding-Layers während des Modeltrainings vorgenommen. Dadurch soll die Reihenfolge der Wörter sowie ihr Kontext besser erfasst werden, um eine höhere Anpassungsfähigkeit an den Datensatz zu ermöglichen. [52]

5 Ansätze

Im Rahmen der Experimente der Arbeit wurden die folgenden Machine- und Deep-Learning-Ansätze verwendet: **Naive Bayes** wurde gewählt, da der Klassifikator sehr gute Ergebnisse bei Klassifikationsproblemen liefert [49]. Auch bei der Klassifikation von Text überzeugt er [25]. Mit den im Bag-of-words- und TF-IDF-Verfahren vektorisierten Testdaten haben wir MultinomialNB und ComplementNB verwendet. Diese Verfahren eignen sich zum einen gut für unbalancierte Daten aber auch für Textklassifizierung [42]. In einer Arbeit zur Sentiment Analyse klassifizierte ComplementNB besser als die restlichen NB Verfahren[19]. Für die Word2Vec- und FastText-Daten (zur Evaluierung der Vektorisierung) haben wir GaussianNB verwendet, um auch die negativen Elemente der vektorisierten Daten verarbeiten zu können. Ebenso wie für die GloVe-Dateien.

Support-Vektor-Maschine (SVM) wurde ausgewählt, da sie sich besonders gut für binäre Klassifizierungsprobleme eignet [44]. Die SVM maximiert den Abstand zwischen den Klassen durch die Suche nach einer optimalen Entscheidungsgrenze, was zu einer hohen Trennschärfe führt [34]. Zudem sind SVM robust gegenüber hochdimensionalen Daten, wie sie häufig bei Textdaten durch Vektorisierung (z. B. TF-IDF) entstehen und optimieren die Balance zwischen precision und recall [28]. Darüber hinaus eignet sich die SVM besonders gut im Falle einer geringen Anzahl von Trainingsbeispielen und ist im Vergleich zu anderen Algorithmen weniger anfällig für Overfitting [6]. Auch in anderen, vergleichbaren Forschungsarbeiten konnte die SVM erfolgreich eingesetzt werden [29, 6, 28].

Ensemble Modelle kombinieren mehrere „schwache Lerner“ um so ein robustes und leistungsstärkeres Modell zu erstellen. Es gibt drei Kategorien: Bagging, Boosting und Stacking, die sich in der Art der Modellkombination unterscheiden. Ensemble-Modelle bieten eine breite Anwendbarkeit auf Fragestellungen des maschinellen Lernens und lassen sich ebenfalls gut für die (Text-)Klassifikation einsetzen, wobei auch mit unbalancierten Daten und Ausreißern gut umgegangen werden kann [26]. Innerhalb der Trainingsphasen wurden Vertreter aus allen drei Kategorien (Bagging: RandomForest; Boosting: XGBoost, LightGBM, CatBoost; Stacking: Kombination von Modellen) trainiert und verglichen.

Recurrent Neural Networks (RNN) speichern durch rekurrente Verbindungen frühere Eingaben und nutzen Kontextinformationen, was sie für Sprachverarbeitung besonders geeignet macht [5, 36]. Eine spezialisierte Form von RNNs sind **Long Short-Term Memory (LSTM)**-Netzwerke und **Gated Recurrent Units (GRU)**, die entwickelt wurden, um die Herausforderungen traditioneller RNNs zu bewältigen. LSTMs lösen das Problem des vanishing gradient, indem sie Speicherzellen und Gate-Mechanismen verwenden. Dadurch können sie langfristige Abhängigkeiten besser erfassen und relevante Informationen gezielt bewahren, was sie besonders leistungsfähig für die Textklassifikation macht [43, 30]. GRUs stellen eine vereinfachte Variante von LSTMs dar und können durch ihre reduzierte Modellkomplexität eine schnellere Berechnung bei vergleichbarer Leistung ermöglichen. GRUs sind insbesondere bei langen Sequenzen

und begrenzten Rechenressourcen effizienter, da sie weniger Parameter haben und schneller trainieren [47]. In mehreren Studien wird die Überlegenheit von LSTM- und GRU-Netzwerken gegenüber klassischen RNNs und traditionellen Machine-Learning-Methoden hervorgehoben [1, 33, 47].

Transformer-Modelle wie BERT erfassen durch Self-Attention lokale und globale Kontexte, was eine präzisere Sprachrepräsentation ermöglicht [41]. Durch bidirektionales Training berücksichtigt BERT sowohl vorherige als auch nachfolgende Wörter, was zu einem besseren Verständnis des Satzkontexts führt [15]. Ein Vorteil von BERT ist das Transfer Learning, bei dem das Modell auf großen Textkorpora vortrainiert wird und mit wenigen gelabelten Daten angepasst werden kann [38]. Im Gegensatz zu klassischen Methoden wie TF-IDF benötigt BERT keine externe Einbettung. Varianten wie RoBERTa optimieren die Trainingsstrategie, während DistilBERT eine ressourcenschonende Alternative darstellt [24]. BERT eignet sich besonders für Hate-Speech-Erkennung und übertrifft klassische ML-Methoden sowie LSTMs und CNNs in der Textklassifikation [50].

6 Experimente

In diesem Kapitel wird das durchgeführte Projekt im Detail beschrieben. Abschnitt 6.1 erläutert den übergeordneten Versuchsaufbau sowie die einzelnen Schritte. Ebenso werden hier die verwendeten Evaluationsmetriken vorgestellt. In Abschnitt 6.3 werden die erzielten Ergebnisse beschrieben und interpretiert.

6.1 Experimentaufbau und Metriken

Das Training erfolgte in 2 Phasen, welche in diesem Kapitel näher erläutert werden sollen. Zuvor soll jedoch auf die für das Training verwendeten Evaluationsmetriken eingegangen werden. Für die Evaluation eines Klassifikationsproblems wie der Erkennung von Hate Speech auf Twitter-Daten können unter anderem Precision und Recall als Metrik verwendet werden. Dabei kann es je nach Anwendungsfall wichtig sein, dass Labels mit besonders hoher Genauigkeit (hohe Precision) oder aber möglichst viele Labels überhaupt (hoher Recall) hervorgesagt werden. Da das Nicht-Erkennen von Hate Speech wie in Kapitel 1 dargelegt schwerwiegendere Konsequenzen nach sich ziehen kann, entschied das Team den Recall auf die gesuchte Klasse als Evaluationsmetrik heranzuziehen. Angesichts des Ungleichgewichts der Klassen im Datensatz wurde zusätzlich der F1-Score verwendet, da er ein harmonisches Mittel aus Präzision und Recall darstellt und somit eine ausgewogene Bewertung der Modellleistung auf der seltenen Zielklasse ermöglicht. Darüber hinaus zeigte [48], dass der F1-Score in Studien mit ähnlicher Zielstellung, die am häufigsten verwendete Metrik ist.

6.1.1 Trainingsphase I

In der ersten Phase wurde auf dem ursprünglichen, unter Kapitel 4.1 beschriebenen Datensatz aus Kaggle trainiert. Wie in Kapitel 4.1 bereits beschrieben ist der entsprechende Testdatensatz nicht gelabelt. Aus diesem Grund wird der Trainingsdatensatz vor jeglichen Verarbeitungsschritten in einen neuen Trainings- und Testdatensatz im Verhältnis 70:30 aufgeteilt. Der neue Testdatensatz dient als finaler Holdout-Datensatz, der erst ab dem u.g. Schritt 9 verwendet wird. So wird sichergestellt, dass keine Erkenntnisse (direkt/indirekt) aus diesen Daten in das Modelltraining einfließen und es beeinflussen.

Naive Bayes: Es wurden die drei Vektorisierungsfunktionen wie in 5 beschrieben zum Training der Modelle genutzt. Je nach verwendetem Naive Bayes Modell (Multinomial, Complement, Gaussian) wurden mittels GridSearch verschiedene Werte für die Parameter für prios, var_smoothing, alpha, fit_prior, class_prior oder norm in den möglichen Kombinationen getestet. Zuletzt wurde diese parametrisierte Suche für die Word2Vec-Vektorisierung noch mit möglichen Kombinationen der Parameter (window, vector_size) der Word2Vec-Funktion getestet. Mit Optimierung der Parameter erzielt Word2Vec ähnliche Werte wie GloVe. Am besten performen die TF-IDF Vektorisierung in Kombination mit MultinomialNB, wobei ComplementNB hier annährend gleiche Werte erzielt und das anhand der GloVe-Vectorizer trainierte Modell. Allerdings erreicht der F1-Score auch für das beste Modell gerade einmal 33%.

Support Vector Machine: Für jedes der drei Vektorisierungsmodelle (tfidf, w2v, GloVe) wurden je vier Kernel (linear, poly, sigmoid, rbf) mit verschiedenen Einstellungen für C, Gamma, Random State und Klassengewicht (um der Unterrepräsentation der Zielklasse zu begegnen) getestet. Beim polynomialen Kernel wurde zudem der Degree variiert. Da die Hyperparameter stark interagieren und die Modellleistung beeinflussen, wurde GridSearchCV eingesetzt, um systematisch alle möglichen Kombinationen zu testen und leistungsstarke Parameterkonfigurationen zu identifizieren. Durch die integrierte Kreuzvalidierung wurde das Trainingsset in k Folds aufgeteilt, wodurch das Modell auf k-1 Folds trainiert und auf dem verbleibenden Fold validiert wird. Dieser Vorgang wurde k-mal wiederholt, was eine robuste Leistungsbewertung und eine Reduzierung von Overfitting ermöglicht. Die Evaluation zeigt, dass random state keine Auswirkungen auf die Modellperformance hat und identifiziert eine Kombination aus tfidf-Vectorizer und rbf-Kernel als bestes Modell.

Ensemble: Für das Training der Ensemble-Modelle auf den ursprünglichen Daten wurden Klassifikatoren aus den Libraries sklearn, xgbm, catboost und lightgbm ausgewählt. Zunächst wurde geprüft, welche der in Kapitel 4.3 beschriebenen Vektorisierungsarten beim Referenzmodell (RandomForest) die besten Ergebnisse liefert – Ergebnis: TF-IDF. Anschließend wurden alle Basismodelle damit trainiert und verglichen. Dabei zeigte sich eine insgesamt schlechte Modellperformance (F1-Score < 64%) sowie große Schwankungen zwischen Precision und Recall, was auf das starke Klassenungleichgewicht und die geringe Datenqualität (Labeling) zurückzuführen sein könnte. Zum Vergleich wurden alle Modelle mit Resampling (SMOTE) erneut trainiert, wodurch sich Precision und Recall zwar annäherten, der F1-Score jedoch niedrig blieb.

Dies bestätigt die Vermutung, dass die Datenqualität unzureichend ist, um präzise Modellvorhersagen zu ermöglichen. Aufgrund der schlechten Performance wurde das Hyperparametertuning nur auszugsweise getestet, blieb aber ohne Verbesserung. Die besten Modelle dieser Phase waren Stacking-/VotingClassifier (Sub-Modelle: SVM, Bayes, RandomForest) und BalancedRandomForest. Der VotingClassifier entfällt jedoch aus der vergleichenden Betrachtung, da der Bezug zu den Submodellen und damit anderen Modellen dieser Phase zu groß ist.

Das **LSTM-Modell** besteht aus einer LSTM-Schicht mit 128 Einheiten, einer Dropout-Schicht (Rate 0,5) und einer abschließenden Dense-Schicht mit Sigmoid-Aktivierung zur binären Klassifikation. Es wurde über zehn Epochen mit einer Batchgröße von 32 trainiert. Zur Berücksichtigung der ungleichen Klassenverteilung kam der Parameter *class_weight* zum Einsatz, um die Minderheitsklasse (Hate Speech) stärker zu gewichten. Für die Vektorisierung wurden GloVe, FastText und Word2Vec getestet. **LSTM mit FastText** erzielte eine sehr hohe Recall-Rate, jedoch bei geringer Präzision – das Modell erkannte viele Hate-Speech-Beispiele, erzeugte aber auch zahlreiche *False Positives*. Die Ergebnisse mit **GloVe** waren insgesamt am ausgewogensten und lieferten die besten Resultate hinsichtlich Genauigkeit und F1-Score. **Word2Vec** schnitt in der Generalisierung am schwächsten ab. Modellanpassungen wie mehr LSTM-Einheiten oder zusätzliche Epochen verbesserten den F1-Score, während eine geringere Dropout-Rate kaum Wirkung zeigte. Ein bidirektionales LSTM erkannte zwar komplexere Muster, senkte jedoch den Recall, da unsichere positive Instanzen häufiger als negativ eingestuft wurden.

Modell	**F1**	**Recall**	**Precision**	**Accuracy**	**MCC**
RNN-GRU, Glove	0.651	0.603	0.706	0.956	0,604
SVC (RBF), TFIDF	0.633	0.610	0.657	0.957	0.610
RNN-LSTM, Glove	0.581	0.638	0.533	0.937	0.55
RNN-LSTM, Fasttext	0.234	0.936	0.134	0.579	0.247
SVC (RBF), Glove	0.572	0.738	0.467	0.933	0.555
BalancedRF, TFIDF	0.555	0.439	0.754	0.953	0.553
ComplementNB, Glove	0.337	0.417	0.282	0.654	0.118
ComplementNB, TFIDF	0.325	0.323	0.326	0.716	0.145

Tabelle 1: Ergebnisse der am besten performenden Modelle (Trainingsphase 1)

Das **GRU-Modell** wurde analog zum LSTM-Modell trainiert, jedoch mit angepassten Parametern, um die architekturspezifischen Eigenschaften von GRUs optimal zu nutzen. Die besten Ergebnisse wurden mit vortrainierten GloVe-Embeddings (200d) erzielt. Durch die geringere Rechenkomplexität konnten 256 GRU-Einheiten verwendet werden, was eine effizientere Verarbeitung der Embeddings ermöglichte. Die maximale Sequenzlänge wurde auf 40 Tokens begrenzt, da GRUs langfristige Abhängigkeiten schwächer modellieren als LSTMs. Ein bidirektionales GRU führte zu starker Überanpassung und schlechteren Testergebnissen. Optimal war ein Spatial-Dropout von 0.35. Die binäre Klassifikation erfolgte über eine Sigmoid-Aktivierung, trainiert wurde über

20 Epochen mit Batchgröße 32. Die Wahl der Hyperparameter erfolgte iterativ und basierte auf umfangreichen empirischen Tests zur Validierungsleistung.

6.1.2 Trainingsphase 2

In der zweiten Trainingsphase wurde auf dem unter Kapitel 4.2 beschriebenen erweitertem Datensatz trainiert. Parallel fand eine Anpassung/Tuning der Hyperparameter statt.

Naive Bayes: Mit dem gemischten Datensatz wurde ebenso vorgegangen wie mit dem ursprünglichen Datensatz in Trainingsphase 1. Es konnte eine deutliche Verbesserung der Performance festgestellt werden, allerdings liegt auch hier der beste F1-Score gerade einmal bei 51%, der Recall konnte hingegen um 40 Prozentpunkte auf etwa 75% gesteigert werden. Vermutlich macht sich hier die bessere Qualität des erweiterten Datensatz bemerkbar. Ferner wurden mit den Parametern(ngram_range, analyzer, norm) der TF-IDF Vektorisierung experimentiert, ohne allerdings Verbesserungen gegenüber den Standardeinstellungen zu erzielen.

Support Vector Machine: Da sich das Modelltraining in Trainingsphase I aufgrund der Vielzahl an Parametern und der damit verbundenen möglichen Konfigurationen mit GridSearch als sehr zeit- und rechenintensiv erwies, wurde in Trainingsphase II — mit dem deutlich umfangreicheren Datensatz — RandomizedSearchCV von sklearn eingesetzt, um vielversprechende Vectorizer-Kernel-Kombinationen effizient zu identifizieren. Als beste Konfiguration wurde so eine Kombination aus tfidf-Vectorizer und linear-Kernel identifiziert. Ïm Vergleich zur Trainingsphase I konnte der Recall leicht verbessert werden. Allerdings verschlechterten sich alle anderen Metriken, was darauf hindeutet, dass das Modell insgesamt mehr Eingaben als Hate Speech klassifiziert, ohne dabei gezielt eine bessere Unterscheidung zu treffen.

Ensemble: Das Vorgehen mit den erweiterten Daten entspricht dem gleichen Vorgehen wie in Trainingsphase 1. Trotz verbesserter Klassenverteilung konnte auch hier wieder eine bessere Performance beim BalancedRandomforestClassifier festgestellt werden, weshalb alle Basismodelle erneut mit veränderten Daten (SMOTE) trainiert wurden. Hierdurch konnte bei vielen Modellen eine geringe Verbesserung erzielt werden. Anschließend wurde versucht, einen BalancedRandomForestClassifier (ohne SMOTE) und einen CatBoostClassifier (mit SMOTE) mittels Hyperparametertuning durch RandomizedSearchCV, optuna sowie eigenen Scorer-Funktionen zu verbessern. Im Ergebnis konnte nur beim CatBoostClassifier der Recall bei gleichbleibendem F1-Score verbessert werden, beim BalancedRandomForestClassifier gabe es keinerlei Verbesserungen.

LSTM: In Trainingsphase 2 wurden die leistungsstärksten Modelle aus Phase 1 erneut mit GloVe-, FastText- und Word2Vec-Vektorisierungen getestet. Aufgrund der erneut schwachen Resultate von FastText und Word2Vec konzentrierte sich die Analyse auf GloVe. Zum Einsatz kam eine speziell für LSTM angepasste GloVe Vektorisierungsfunktion auf Basis des Keras-Tokenizers, fester Sequenzlängen und eines Wortindexbasierten Vokabulars, wodurch die Texte in strukturierte Wortvektorsequenzen über-

führt wurden. Das Modelltraining erfolgte auf einem erweiterten Datensatz unter Einsatz von Early Stopping und ReduceLROnPlateau zur Stabilisierung und Vermeidung von Overfitting. Modell1 trainiert über zehn feste Epochen mit hoher Dropout-Rate (0.5), jedoch ohne Lernratenanpassung. Modell2 ergänzt diese um Early Stopping, ReduceLROnPlateau und bis zu 50 Epochen, was eine bessere Anpassung ermöglicht. Modell3 entspricht Modell2, verwendet jedoch eine höhere Dropout-Rate (0.5 statt 0.35) zur stärkeren Regularisierung. Modell2 erzielte insgesamt die stabilste Performance mit der höchsten Recall-Rate.

GRU Die zweite Trainingsphase für das GRU-Modell folgte einem ähnlichen Aufbau wie beim LSTM. Auch hier wurden die besten Modelle der ersten Phase mit GloVe, FastText und Word2Vec getestet, wobei erneut GloVe die besten Ergebnisse lieferte. FastText und Word2Vec blieben leistungsschwach und wurden ausgeschlossen. Auf dem neuen gemischten Datensatz wurde erneut versucht, die Modellleistung durch erweitertes Hyperparameter-Tuning zu steigern. Dabei zeigte sich jedoch, dass die Architektur und Parametrisierung aus Phase I bereits das beste Ergebnis lieferten, sodass keine weiteren Anpassungen vorgenommen wurden. In dieser Phase II wurden außerdem die Schritte zur Behandlung des Ungleichgewichts im Datensatz, wie der Einsatz von Oversamplern und Klassengewichten, bewusst weggelassen, da der neue Datensatz kein starkes Ungleichgewicht mehr aufwies.

BERT/RoBERTa: Zum Finetuning von LLMs wurden drei Modelle ausgewählt und gleichermaßen auf den neuen Daten nachtrainiert: bert-based-uncased ([16]), roberta-base-sentiment ([12]) und roberta-base-hate ([9]). Hierbei ist besonders hervorzuheben, dass das RoBERTa Hate-Modell bereits auf einem Teil der neuen Daten trainiert wurde. Um die Trainingszeit zu verkürzen wurde auszugsweise Training mit LoRA versucht. Da jedoch kein nennenswerter Zeitgewinn zustande kam, wurde dies nicht weiter angewendet. Um das roberta-sentiment-Modell mit den anderen Modellen vergleichen zu können, wurde die Ausgabe auf eine binäre Klassifikation nachtrainiert. Im Vergleich der Modelle zeigte sich, dass durch das Nachtraining eine bessere Genauigkeit der Modelle erreicht wird, diese jedoch nicht gravierend besser ist als die der vorher betrachteten Modelle.

Modell	**F1**	**Recall**	**Precision**	**Accuracy**	**MCC**
RoBERTa	0.587	0.561	0.614	0.833	0.483
Bert	0.569	0.535	0.608	0.831	0.465
BalancedRF TFIDF	0.564	0.702	0.471	0.771	0.432
RNN-GRU, Glove	0.563	0.602	0.529	0.804	0.442
SVC (Linear), TFIDF	0.539	0.779	0.412	0.720	0.402
RNN-LSTM, Glove	0.524	0.803	0.389	0.693	0.383
ComplementNB, TFIDF	0.513	0.754	0.389	0.699	0.364

Tabelle 2: Ergebnisse der am besten performenden Modelle (Trainingsphase 2)

6.2 Zusätzliche Evaluation auf manuell gelabelten Daten

Wie bereits in Kapitel 4.1 erwähnt, konnten aus Sicht des Teams inkonsistent bzw. fraglich gelabelte Datensätze identifiziert werden. Aus diesem Grund, sowie vor dem Hintergrund der mittelmäßigen Modellperformance aus Trainingsphase 2, wurde eine zusätzliche Evaluation durchgeführt. Dazu wurden 430 Datensätze aus den finalen Testdaten manuell überprüft, um sicherzustellen, dass ihre Labels mit der festgelegten Definition von Hate Speech übereinstimmen. In diesem Zuge wurden insgesamt 69 Labels angepasst. Hervorzuhebende Beispiele für die Änderung zu Hate Speech sind u.a. *„You must kill all non-believers."* und *„California would be a better place without all the dirty mexicans"*, wohingegen *„The world would be a very dull place if we were consigned to only having friends with matching genitalia."* und *„Yeah! Give those homophobes a new language to learn!"* nicht mehr als Hate Speech klassifiziert wurden.

Modell	**F1**	**Veränderung**	**Recall**	**Veränderung**
RoBERTa	0.847	+0.068	0.814	+0.046
Bert	0.781	+0.043	0.797	+0.023
RNN-GRU	0.765	+0.028	0.734	+0.008
RNN-LSTM	0.739	+0.038	0.907	+0.019
LinearSVC	0.753	+0.031	0.785	+0.012
BalancedRF	0.736	+0.053	0.701	+0.034
ComplementNB	0.729	+0.054	0.921	+0.040

Tabelle 3: Ergebnisse der Modelle auf händisch gelabelten Daten

In der Tabelle 3 sind jeweils die Veränderungen der Metriken zu der Klassifikation der Stichprobe ohne Veränderung des Labels aufgeführt. Im Vergleich mit den Ergebnissen aus der zweiten Trainingsphase zeigen sich zwar bei der Stichprobe verbesserte Metriken, diese werden bei den überprüften Labeln jedoch noch weiter verbessert. Die Detailbetrachtung der hier falsch vorhergesagten Tweets bestätigt die Ergebnisse aus Trainingsphase 2: auch hier werden Tweets (jedoch im Verhältnis weniger als in Trainingsphase 2) von den Modellen als Hate Speech deklariert, in denen Personengruppen und Beleidigungen vorkommen, wobei es sich jedoch nicht um Hate Speech gemäß Definition handelt. Des Weiteren werden solche Tweets falsch klassifiziert, die besonders lang sind und bei denen sich der Inhalt nicht nur auf Beleidigungen beschränkt.

6.3 Beschreibung und Interpretation der Ergebnisse

Innerhalb von beiden Trainingsphasen konnten jeweils nur mittelmäßige Ergebnisse im Bezug zu den gewählten Metriken erreicht werden. In der Detailauswertung der Modelle der zweiten Trainingsphase zeigte sich, dass bei den Testdaten ca. 5.6% von allen Modellen und ca. 29% von mindestens 4/7 Modellen falsch klassifiziert werden, wobei BERT sowie RoBERTa verhältnismäßig besser Vorhersagen getroffen haben. Dies könnte daran liegen, dass diesen Modellen ein komplexeres Sprachverständnis möglich ist

und somit der Inhalt und Kontext von Beleidigungen besser erfasst werden können. Eine weiterführende Analyse der mehrheitlich falsch klassifizierten Tweets zeigt, dass in vielen Fällen zwar keine Hate Speech vorliegt, jedoch Beleidigungen oder Begriffe für Personengruppen enthalten sind. Dies stützt die Annahme, dass viele Modelle Schwierigkeiten haben, zwischen allgemeinen und gruppenbezogenen Beleidigungen zu differenzieren. Zudem wurde im Rahmen einer gezielten Stichprobe die Datenqualität kritisch hinterfragt und evaluiert, wie die Modelle auf Tweets reagieren, die aus Teamsicht klar als Hate Speech einzuordnen sind oder nicht. Da Sprache und insbesondere komplexe Phänomene wie Hate Speech nur schwer objektiv binär zu labeln sind, kann ein Bias in den neuen Labels nicht ausgeschlossen werden. Die daraus abgeleiteten Metriken dienen daher nicht der Bewertung der Modellgüte, sondern lediglich dem Vergleich. Eine Limitation dieser Arbeit ist der nur eingeschränkt mögliche direkte Vergleich der beiden Trainingsphasen, da diese auf unterschiedlichen Testdatensätzen evaluiert wurden.

7 Ausblick

Ein zentrales Problem war die Datenqualität, insbesondere aufgrund der mangelnden Eindeutigkeit des Begriffs „Hate Speech“, was eine klare Unterscheidung zwischen Hate Speech und Nicht-Hate Speech erschwerte. Eine vollständige manuelle Korrektur der Labels und damit einhergehende Konsistenz wäre womöglich hilfreich gewesen, konnte jedoch aus Zeitgründen nicht realisiert werden. Zwar wurde in der Endphase ein kleiner, manuell gelabelter Testdatensatz erstellt, dieser ist jedoch nicht groß genug für eine aussagekräftige Bewertung. Zukünftig könnte ein qualitativ hochwertiger, ggf. eigenständig erfasst und kontrollierter Datensatz zu einer besseren Modellleistung führen. Zudem wäre es vielversprechend, Hashtags und Emojis nicht nur als Textbestandteile, sondern als eigenständige Features im Modell zu nutzen, um zusätzliche semantische Informationen zu integrieren.

8 Zusammenfassung und Fazit

Im Rahmen der Arbeit wurde versucht Hate Speech in Tweets mithilfe von klassischen maschinellen Lernverfahren sowie Deep-Learning- und Transformer-Modellen zu identifizieren. Dabei konnten über verschiedene Modelle und Trainingsdatensätze hinweg nur moderate Testergebnisse erzeugt werden. Besonders klassische Verfahren wiesen ein unzureichendes Sprachverständnis auf, um komplexe Konzepte wie Hate Speech korrekt zu klassifizieren. Vor allem in der Einzelbetrachtung konnte festgestellt werden, dass zwar Beleidigungen von Einzelpersonen erkannt werden können, jedoch nur schwer zwischen der für Hate Speech ausschlaggebenden Verbindung einer Beleidigung zu Merkmalen wie Nationalität, Religion oder Sexualität fehlerfrei unterschieden werden kann und vor allem hier Fehlklassifikationen entstehen.

Literatur

[1] Saheed Salahudeen Abdullahi, Sun Yiming, and Shamsuddeen Hassan Muhammad et al. Deep sequence models for text classification tasks. *arXiv preprint arXiv:2207.08880*, 2022.

[2] Sindhu Abro, Sarang Shaikh, Zahid Hussain Khand, Ali Zafar, Sajid Khan, and Ghulam Mujtaba. Automatic hate speech detection using machine learning: A comparative study. *International Journal of Advanced Computer Science and Applications*, 11(8), 2020.

[3] Stephen Akuma, Tyosar Lubem, and Isaac Terngu Adom. Comparing bag of words and tf-idf with different models for hate speech detection from live tweets. *International Journal of Information Technology*, 14(7):3629–3635, 2022.

[4] Isaac Terngu Adom Akuma Stephen, Tyosar Lubem. Comparing bag of words and tf-idf with different models for hate speech detection from live tweets. *International Journal of Information Technology*, 2022.

[5] Md. Zahangir Alom, Tarek M. Taha, and Christopher Yakopcic et al. Recurrent neural networks: A comprehensive review of applications and research directions. *MDPI Information*, 15(9):517, 2023.

[6] Anugraha Antoo Kanjookaran Binil Tom Jose Anagha Abraham, Antony J Kolanchery and Dhanya PM. Hate speech detection in twitter using different models. *ITM Web Conf. Volume 56, 2023*, 2023.

[7] analyticsvidhya. Analytics vidhya homepage. *analyticsvidhya*, 2024.

[8] analyticsvidhya. Ultimate guide to deal with text data (using python) – for data scientists and engineers. *analyticsvidhya*, 2024.

[9] Dimosthenis Antypas and Jose Camacho-Collados. Robust hate speech detection in social media: A cross-dataset empirical evaluation. In *The 7th Workshop on Online Abuse and Harms (WOAH)*, pages 231–242, Toronto, Canada, July 2023. Association for Computational Linguistics.

[10] Emma Holkeri Matti Näsi Pekka Räsänen Atte Oksanen, James Hawdon. Exposure to online hate among young social media users. 18, 2014.

[11] Yıldız Aydın. Sentiment analyzing from tweet data's using bag of words and word2vec. *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, 12(2):412–417.

[12] Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online, November 2020. Association for Computational Linguistics.

[13] David R. Williams Geneene N. Thompson Brendesha M. Tynes, Michael T. Giang. Online racial discrimination and psychological adjustment among adolescents. *Journal of Adolescent Health*, 43(6), 2008.

[14] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515, 2017.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[17] Dictionary Cambridge. definition: hate speech, 23.03.2025.

[18] geeksforgeeks. Python – lemmatization approaches with examples. *geeksforgeeks*, 2022.

[19] Nur Hayatin, Suraya Alias, Po Hung Lai, and Mohd Shamrie Sainin. Sentiment analysis based on probabilistic classifier techniques in various indonesian review data. *Jordanian Journal of Computers and Information Technology*, 8(3), 2022.

[20] IBM. Was ist stemming?, 03.03.2025.

[21] Carlo Schwarz Karsten Müller. Fanning the flames of hate: Social media and hate crime. *Journal of the European Economic Association*, 19(4), 2021.

[22] Chris J Kennedy, Geoff Bacon, Alexander Sahn, and Claudia von Vacano. Constructing interval variables via faceted rasch measurement and multitask deep learning: a hate speech application. *arXiv preprint arXiv:2009.10277*, 2020.

[23] Sudalai Raj Kumar. Getting started with text preprocessing. *kaggle*, 2019.

[24] Yinhan Liu, Myle Ott, Naman Goyal, Du Jingfei, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach.

[25] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naive bayes-which naive bayes? In *CEAS*, volume 17, pages 28–69. Mountain View, CA, 2006.

[26] Domor Mienye and Yanxia Sun. A survey of ensemble learning: Concepts, algorithms, applications, and prospects. *IEEE Access*, PP:1–1, 09 2022.

[27] Zewdie Mossie, Jenq-Haur Wang, et al. Social network hate speech detection for amharic language. *Computer Science & Information Technology*, pages 41–55, 2018.

[28] Rani Kurnia Putri Muhammad Athoillah1. Utilizing support vector machines to detect hate speech on social media. *Science, Engineering and Technology*, 2024.

[29] Adianto Jeremy Wijaya Nina Sevani, Iwan A. Soenandi. Detection of hate speech by employing support vector machine with word2vec model. *2021 7th International Conference on Electrical, Electronics and Information Engineering (ICEEIE)*, 2021.

[30] Jakub Nowak, Ahmet Taspinar, and Rafal Scherer. Lstm recurrent neural networks for short text and sentiment classification. *ResearchGate*, 2017.

[31] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[32] Holkeri Emma Keipi Teo Näsi Matti Oksanen Atte Räsänen Pekka, Hawdon James. Targets of online hate: Examining determinants of victimization among young finnish facebook users. *Violence and Victims*, 31(4), 2016.

[33] H. Sak, A. Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *INTERSPEECH*, 2014.

[34] Stefan Selle. Data science training - supervised learning: Lineare klassifizierer. 2024.

[35] Aditya Sharma and Alex Daniels. Tweets sentiment analysis via word embeddings and machine learning techniques. *arXiv preprint arXiv:2007.04303*, 2020.

[36] Farhad Mortezapour Shiri, Thinagaran Perumal, Norwati Mustapha, and Raihani Mohamed. A comprehensive overview and comparative analysis on deep learning models: Cnn, rnn, lstm, gru. *arXiv preprint*, 2023.

[37] Anirudha Simha. Understanding tf-idf for machine learning. *capitalone*, 2021.

[38] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification?

[39] Atte Oksanen Pekka Räsänen Teo Keipi, Matti Näsi. Online hate and harmful content, cross-national perspectives. page 154, 2016.

[40] Ali Toosi. Twitter sentiment analysis. *kaggle*, 2019.

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

[42] Merve Veziroğlu, Erkan Veziroğlu, and İhsan Ömür Bucak. Performance comparison between naive bayes and machine learning algorithms for news classification. In *Bayesian Inference-Recent Trends*. IntechOpen, 2024.

[43] Zhongwei Wan. Text classification: A perspective of deep learning methods. *arXiv preprint arXiv:2309.13761*, 2023.

[44] Qiyu Wang. Support vector machine algorithm in machine learning. *2022 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, 2022.

[45] Zeerak Waseem and Dirk Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California, June 2016. Association for Computational Linguistics.

[46] Mikołaj Winiewski Wiktor Soral, Michał Bilewicz. Exposure to hate speech increases prejudice through desensitization. *Aggressive Behavior*, 44(2), 2018.

[47] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing.

[48] Sabrina Tiun Zainab Mansur, Nazlia Omar. Twitter hate speech detection: A systematic review of methods, taxonomy analysis, challenges, and opportunities. *IEEE Access*, 2023.

[49] Harry Zhang. The optimality of naive bayes. *Aa*, 1(2):3, 2004.

[50] Rong Zhang, Revanth Gangi Reddy, Md Arafat Sultan, Vittorio Castelli, Anthony Ferritto, Radu Florian, Efsun Sarioglu Kayi, Salim Roukos, Avirup Sil, and Todd Ward. Multi-stage pre-training for low-resource domain adaptation.

[51] Dino Zivojevic. Capturing morphological structure in word embeddings: Word2vec vs fasttext. *atlantbh*, 2024.

[52] Enes Zvornicanin. What are embedding layers in neural networks? *Baeldung*, 2024.

Erklärung

Wir erklären, dass wir die schriftliche Ausarbeitung zum Seminar selbstständig und ohne unzulässige Inanspruchnahme Dritter verfasst haben. Wir haben dabei nur die angegebenen Quellen und Hilfsmittel verwendet und die aus diesen wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht. Die Versicherung selbstständiger Arbeit gilt auch für enthaltene Zeichnungen, Skizzen oder graphische Darstellungen. Die Ausarbeitung wurde bisher in gleicher oder ähnlicher Form weder derselben noch einer anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Mit der Abgabe der elektronischen Fassung der endgültigen Version der Ausarbeitung nehmen wir zur Kenntnis, dass diese mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiate geprüft werden kann und ausschließlich für Prüfungszwecke gespeichert wird.

30.03.2025 Engels Friedrich Ntali Tuychieva Wall

(Ort, Datum) (Unterschrift)