Erweiterung der BERT-Architektur für die Klassifikation von Hassrede

Félix Fautsch
Fakultät für Informatik und Mathematik
Hochschule München University of Applied Sciences
München, Deutschland
felixfautsch@gmail.com

***Abstrakt*— Transformer-Modelle haben sich in den letzten Jahren als Standard in vielen NLP-Aufgaben etabliert. Diese Arbeit untersucht, inwiefern Erweiterungen der BERT-Architektur die Leistung bei der automatisierten Klassifikation von Hassrede verbessern können. Unsere Forschung hat gezeigt, dass unterschiedliche Erweiterungen, wie etwa CNNs oder GCNs, bessere Ergebnisse liefern als Standard Varianten. Unsere Modelle haben unter anderem S-Scores von 0.6306 (CNN) und Recall Werte von 0.9216 (SAGEConv) erzielt.**

Keywords—Natural Language Processing, BERT, CNN, GCN, Hassrede, Textklassifikation, machine learning

I. Einleitung

Die Verarbeitung natürlicher Sprache (engl. natural language processing) ist ein Teilgebiet des machine learnings, das sich mit der computergestützten Verarbeitung menschlicher Sprache befasst, mit dem Ziel, Maschinen in die Lage zu versetzen, menschliche Sprache zu verstehen und selbst zu erzeugen (Acheampong et al., 2021). Es handelt sich dementsprechend um ein interdisziplinäres Forschungsfeld, das unter anderem Methoden der Informatik und Linguistik kombiniert, um Maschinen die Verarbeitung und das Verständnis menschlicher Sprache zu ermöglichen. Eine Aufgabenstellung aus diesem Forschungsgebiet ist die Klassifikation von Texten. Die Aufgabe besteht darin, einer gegebenen Textsequenz, sprich einem Satz, vordefinierte Kategorien zuzuordnen (Sun et al., 2020). Eine Textklassifikation kann je nach Zielsetzung unterschiedliche Spezialisierungen annehmen, etwa die Detektion von Hassrede oder die Analyse von Stimmungslagen. Unsere Forschung beschäftigt sich mit der automatisierten Klassifikation von Hassrede. Dabei soll ein ausgewähltes Modell des maschinellen Lernens durch gezieltes Finetuning so optimiert werden, dass es in der Lage ist, Äußerungen wie etwa “Dummköpfe raus, also schleich dich!” als Hassrede zu klassifizieren. Was für uns Menschen relativ leicht erscheint, stellt für Computer eine Herausforderung dar. In den letzten Jahren hat die Forschung in diesem Bereich massive Fortschritte erzielt. Neben long-short-term-memory Modellen, logistischen Regressionen, Random Forest Ansätzen oder Modellen die auf neuronalen Netzen basieren steht insbesondere das 2018 entwickelte Modell BERT im Fokus der Forschung (Bai, 2018; Devlin et al., 2019; Lu et al., 2020; Shah et al., 2020; Zhou et al., 2015). Der Vorteil von BERT (Bidirectional Encoder Representations from Transformers) ist, dass das Modell auf einer bidirektionalen Architektur (Transformers) basiert (Devlin et al., 2019). Dies bedeutet, dass Tokens, anstatt nur von links nach rechts, von beiden Seiten gleichzeitig im Kontext analysiert werden, wodurch ein tieferes Verständnis sprachlicher Zusammenhänge ermöglicht wird (Aftan & Shah, 2023). Da BERT in seiner Standardform (bsp.: bert-base) für die maskierte Wortvorhersage und die Vorhersage des nächsten Satzes auf Klartext trainiert ist, muss man das Modell auf die Texklassifikation finetunen. Die einfachste Form ist dabei die Verwendung des finalen hidden state h des CLS Tokens, welcher eine Zusammenfassung des Textes enthält. Die Architektur wird um eine finale Schicht mit einer simplen Softmax erweitert, welche auf Basis der gefinetunten Gewichte W und der Textzusammenfassung h die Wahrscheinlichkeit für die Klasse c vorhersagt (Sun et al., 2020).

$$p(c|\mathbf{h}) = \text{softmax}(W\mathbf{h}) \tag{1}$$

Diese einfache Anpassung von BERT, zusammen mit einem gelabelten Datensatz, erzeugt in den meisten Fällen bereits vielversprechende Ergebnisse. BERT alleine hat jedoch auch Nachteile. Neben overfitting Problemen bei kleinen Datensätzen existiert oft ein Mangel an lokalen Merkmalsextraktionen. Durch die bidirektionale Transformer Architektur werden globale Kontexte gut erfasst, jedoch gehen dadurch oft lokale Muster verloren (Kovaleva et al., 2019). Obwohl die meisten Deep Learning Netzwerke wie CNNs durch bei der Kodierung weitreichender Abhängigkeitsinformationen des Textes eingeschränkt sind, zeichnen diese sich im Gegensatz zu BERT im Bereich der lokalen Kontexte durch die Einbettung semantischer und syntaktischer Informationen in eine gelernte Darstellung gut aus (Lu et al., 2020). Somit würde eine Kombination der Stärken von BERT und ausgewählten neuronalen Netzen in einem einzigen Modell theoretisch ein leistungsfähigeres Modell erzeugen. Ziel dieser Arbeit ist es, die Möglichkeiten der Erweiterung der BERT-Architektur systematisch zu untersuchen und durch gezielte Experimente weiterentwickelte Modellvarianten zu evaluieren, um eine fundierte Basis für zukünftige Forschungsansätze zu schaffen.

II. Methodik

Die für diese Forschung verwendete Methodik lässt sich in zwei Phasen aufteilen. In der ersten Phase wird eine Literaturrecherche zum Thema “Erweiterung der BERT-Architektur” durchgeführt. Ziel dieser strukturierten Recherche ist es, Ansätze in der Literatur zu identifizieren, die sich mit der Erweiterung von BERT Modellen beschäftigen. Hier orientieren wir uns grob an der Methodik von Kitchenham et al. (2002). In der zweiten Phase entwickeln wir die Modelle und führen gezielte Experimente durch. Hiebei handelt es sich um die Erstellung von Prototypen. Dies bildet den technischen Aspekt dieser Forschungsarbeit ab. Dabei ist zu bemerken, dass neben den in der Literatur identifizierten Ansätzen auch eigene, während Experimenten aufgedeckte Erweiterungen eingeführt werden. Der Fokus liegt dabei auf einer hohen Nachvollziehbarkeit, weswegen wir den Quellcode über Github zur Verfügung stellen. Nach einer Beschreibung des verwendeten Datensatzes definieren wir die einzelnen erweiterten Architekturen und skizzieren den

Ablauf der Experimente. Im Anschluss folgt eine Datenanalyse mit abschließender Evaluation der Modelle.

III. Literaturrecherche

Bei der Literaturrecherche liegt die Motivation darin, den aktuellen Stand des Forschungsgebiets offen darzulegen. Durch die Identifikation aller der bereits angewendeten Erweiterungen der BERT-Architektur ermöglichen wir ein grundlegendes Verständnis über die behandelte Thematik. Die Literaturrecherche wurde dabei in zwei Iterationen durchgeführt. In einer initialen Iteration haben wir das Forschungsgebiet untersucht und eine grobe Taxonomie erzeugt. Die identifizierten Kategorien, beruhend auf den möglichen Ansätzen zur Erweiterung von BERT, haben die zweite Phase gesteuert. Innerhalb der jeweiligen Klassifikationen wurden somit sämtliche relevanten Forschungsarbeiten identifiziert. Dabei lag der Fokus auf Ansätzen, die sich mit Klassifikationsaufgaben beschäftigen. Bei der identifizierten Literatur handelt es sich um keine vollständige Aufzählung. Bei den möglichen Erweiterungen aus der Literatur handelt es sich um convolutional neural networks (CNN), graph neural/ convolutional networks (GNN/GCN), fully connected networks (FC), recurrent neural networks (RNN) und long-short-term-memory (LSTM). Die Erklärung und Funktion sowie deren Vorteile folgt in einem späteren Kapitel. Uns ist bewusst, dass LSTM und RNN aus der gleichen Familie von neuronalen Netzen stammen. Im Rahmen unserer Forschung betrachten wir beide dennoch getrennt. Neben der Klassifikation von Stimmungen sowie Sarkasmus behandeln einzelne Forschungsartikel explizit die Klassifikation von Hassrede. Dabei verwenden die meisten Ansätze BERT mit einer Erweiterung an CNN-Schichten (Putra & Wang, 2024; Safaya et al., 2020). CNNs kommen auch im Bereich der Klassifikation von Depressionen in Texten zum Einsatz. Neben CNN oder RNN Zusätzen existieren zahlreiche Forschungen zur Einbettung von Graphen und somit von GNNs beziehungsweise GCNs in die BERT Architektur. Hierbei werden meist hybride Architekturen, ähnlich wie beim Ensemble Learning, vorgestellt. Unsere Forschung richtet sich an den fünf identifizierten Kategorien von Erweiterungen.

Erweiterung	Literatur
CNN	(Kaur & Kaur, 2023); (Rodrigues Makiuchi et al., 2019); (R. Abas et al., 2022); (Dong et al., 2020); (Wan & Li, 2022); (Safaya et al., 2020); (Alghamdi et al., 2022); (Putra & Wang, 2024); (Beniwal & Saraswat, 2024); (Alyoubi et al., 2023); (Chen et al., 2022); (Xin & Zakaria, 2024); (Zheng & Yang, 2019)
GNN/GCN	(Lu et al., 2020); (Zhao et al., 2024); (Perin et al., 2025); (Yang & Cui, 2021); (Taggu & Teyi, 2025); (Jin & Zhao, 2024); (Roethel et al., 2023); (Huang et al., 2022)
FC	(Gao et al., 2019); (P. Liu et al., 2021)
RNN	(S et al., 2022); (Eang & Lee, 2024); (PBV et al., 2025); (Saraswat & Beniwal, 2024)
LSTM	(Pandey & Singh, 2023); (Muralitharan & Arumugam, 2024); (H. Liu & Xie, 2021)

Tabelle 1: Identifizierte Literatur

IV. Datensatz

Bei der automatisierten Klassifikation von Hassrede handelt es sich meistens um eine binäre Text Klassifikation. Texte sollen dabei entweder als Hassrede oder als Nicht-Hassrede klassifiziert werden. Im Rahmen unserer Forschung verwenden wir den HOCON34k Datensatz. Der Datensatz besteht aus 34.223 Kommentaren in deutscher Sprache im Zusammenhang mit dem öffentlichen Diskurs in deutschen Zeitungen (Keller et al., 2025). Jeder Kommentar ist mit einem binären Label versehen: 1 für Hassrede und 0 für Nicht-Hassrede. Der Datensatz beinhaltet bereits vorgesehene Aufteilungen, wobei die präzisere Aufteilung Annotationen von zwölf Moderatoren und die größere Aufteilung von neunundzwanzig Moderatoren enthält. Wir verwenden für unsere Experimente den vorgegebenen größeren train-test-val Split, sprich *split_all*. In der Vorverarbeitung haben wir Duplikate entfernt. Da unsere Experimente mehrere Variablen enthalten, versuchen wir den Datensatz so einzusetzen, dass er für anschließende Forschungen leicht reproduzierbar ist. Hierfür haben wir die einzelnen Texte bereinigt, indem wir @, #, html-tags, Leer- und Sonderzeichen entfernt haben. Zum Trainieren der einzelnen Modelle haben wir den Trainingsdatensatz nicht oversampled.

Abbildung 1: WordCloud des HOCON34k Datensatzes

Dies ist darauf zurückzuführen, dass ein random oversampling nicht immer reproduzierbar ist. Da wir einen hohen Wert auf die Nachvollziehbarkeit legen, verwenden wir stattdessen die Klassengewichte. Dabei teilt sich der Datensatz bei Verwendung der größeren Aufteilung folgendermaßen auf: 0.84714 % Nicht-Hassrede zu 0.15286 % Hassrede. Die dazugehörigen Klassengewichte, als torch tensor, nehmen dementsprechend folgende Form an: [0.5902, 3.2710]. Die Abbildung eins stellt die Herausforderung der automatisierten Klassifikation von deutscher Hassrede dar. Die Wordcloud zeigt die 200 häufigsten Wörter des Datensatzes nach der Entfernung von jeglichen Stoppwörtern. Die Größe der Wörter ist representative für die jeweilige Häufigkeit des Wortes. Die WordCloud enthält, bis auf einen Ausreißer, keine direkten Schimpfwörter. Aus diesem Grund ist die automatisierte Detektion von Hassrede kompliziert. Es reicht nicht bloß, Schimpwörter zu erkennen, sondern man muss die einzelnen Sätze in ihrem Kontext betrachten. Dabei spielt vor allem die Erkennung von indirekter Hassrede eine zentrale Rolle, um effiziente Modelle zu entwickeln.

V. Architekturen

In diesem Kapitel stellen wir die von uns implementierten Architekturen vor und gehen kurz auf deren Vor und Nachteile ein.

A. CNN

Die convolutional layers wurden erstmals eingeführt, um im Bereich der digitalen Bildverarbeitung visuelle Muster zu erkennen, werden seitdems jedoch auch für andere Aufgaben, wie beispielsweise die Verarbeitung von Texten, eingesetzt. Die Architektur der BERT-CNN Modelle folgt meist der gleichen Struktur. Da es sich bei den Daten um Texte, beziehungsweise um Embeddings handelt, verwenden die Modelle conv1D Schichten. Bei diesen Schichten handelt es sich um eindimensionale CNN Schichten.

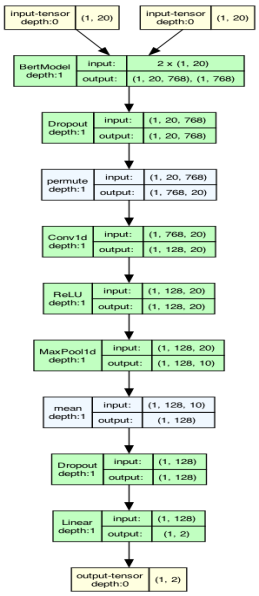

Abb. 2: Standard BERT-CNN Architektur

Die conv2D Schichten sind für zweidimensionale Daten wie Bilder geeignet. Der convolutional layer ist immer von einer Pooling Schicht gefolgt. In unserer Architektur verwenden wir dafür MaxPooling. Die Ausgaben der MaxPooling Schicht werden in einem finalen, fully connected layer, zu Logits verschafft, sodass wir am Ende einen Wert für die beiden Klassen haben. Die Auswahl der Größe der Filter für die convolutionelle Operationen auf den Embeddings zum Erstellen präziser Merkmalskarten (engl. feature maps) ist wichtig, um den Fokus des Modells auf die lokalen Merkmale der Embeddings zu richten. Je nach Forschung werden hier unterschiedliche Filter eingesetzt. Convolutional Neural Networks werden vor allem in Kombination mit BERT verwendet, um die lokalen Merkmale zu extrahieren, was zu einer Reduzierung der Dimension der Daten führt (Kaur & Kaur, 2023). Aus diesem Grund vermeiden wir in unserem Modell die Verwendung von größeren Kernels (> 3). Die standardisierte Architektur unseres BERT-CNN Modells folgt dem Aufbau aus Abbildung zwei. Eine einzelne convolutionelle Schicht berechnet eine convolutional Funktion anhand eines 3x3 Filters mit einem Padding von eins auf den semantischen Vektoren. Nach einer ReLu (engl. rectified linear unit) Aktivierungsfunktion setzen wir MaxPooling mit einem Kernel von zwei ein, um die lokalen Merkmale so gut wie möglich zu behalten. Einzelne Dropout Schichten werden verwendet, um das Modell zu regularisieren und ein mögliches Overfitting zu vermeiden. Bei den Hyperparametern orientieren wir uns allgemein an der Forschung von R. Abas et al. (2022). Die einzelnen Hyperparameter werden im Kapitel der Ergebnisse aufgeführt. Neben dem Modell aus Abbildung zwei implementieren wir simultan ein BERT-CNN Modell, mit drei anstelle von einer CNN Schicht.

B. FC

In der Darstellung der Architektur des BERT-CNN Modells wurde am Ende, wie bei allen CNNs, eine lineare Schicht eingefügt, um die Outputs der CNN Schicht zu verarbeiten und so die Logits der jeweiligen Klassen zu erhalten. Bei dieser linearen Schicht handelt es sich um eine fully connected Schicht (FC). In einem FC ist jeder Neuron mit den Neuronen der Schicht von davor verbunden. Diese führt eine lineare Transformation der Daten durch. Bei mehreren Schichten wird meistens auch noch eine nicht-lineare Aktivierungsfunktion angewendet. Jedes Neuron berechnet ein Gewicht und einen Bias. Diese werden während des Trainings angepasst, sodass das Modell am Ende die besten Ergebnisse liefert. Wir implementieren ein feed forward Netzwerk, welches ähnlich wie das BERT-CNN die Outputs von BERT als Eingabe verwendet. Entscheidend dabei sind die Aktivierungsfunktionen. Im Kontext unserer Experimente testen wir mehrere Aktivierungsfunktionen, unter anderem ReLu, LeakyReLu, Sigmoid (um die Leistung sehr einfacher Modelle zu überprüfen) und Tanh (gleiche Aktivierungsfunktion wie bei BERT). Bei der Anzahl an FC Schichten haben wir uns initial für vier entschieden, um die Komplexität zu minimieren.

C. RNN

Die normalen feed forward Netzwerke aus dem vorherigen Abschnitt leiden unter einigen Problemen. Obwohl sie äußert schnell sind, fällt es reinen FC Schichten schwierig, sequentielle Daten zu verarbeiten und berücksichtigen vorherige Werte nicht. Hier kommen RNNs ins Spiel (S et al., 2022). Recurrent neural Netzwerke haben einen großen Vorteil im Gegensatz zu feed forward Netzwerken, da sie zu einer Klasse von neuronalen Netzen gehören, bei denen die Ausgaben der vorherigen Schicht als Eingabe in die aktuelle Schicht einfließen und Informationen aus der Vergangenheit somit weitergegeben werden (S et al., 2022). Dies bedeutet, dass RNNs in der Lage sind, zeitliche Abhängigkeiten zu modellieren. Dies beruht auf dem Konzept des Speichers, der Informationen über frühere Berechnungen bis zu einem bestimmten Zeitpunkt speichert (S et al., 2022). Da wir simple RNNs implementieren, besteht unter anderem das Problem der Verschwindenden Gradienten (engl. vanishing gradients). Aus diesem Grund verwenden wir innerhalb der einfachen BERT-RNN Architektur Aktivierungsfunktionen, die spezifisch für dieses Problem entwickelt wurden, LeakyReLu und Gelu (Gaussian Error Linear Unit). RNNs haben ebenfalls Probleme mit langen Sequenzen, wobei wichtige Informationen nicht berücksichtig werden. Aus diesem Grund wurden LSTMs als eine modernere Variante von herkömmlichen RNNs entwickelt. Diese sind theoretisch besser geeignet für unsere Forschung, weshalb wir sie ebenfalls implementieren und im Abschnitt *E* definieren.

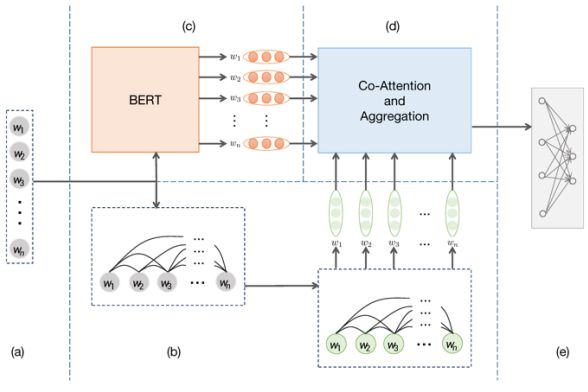

Abb. 3: BERT-GNN Ansatz (Yang & Cui, 2021)

D. GNN/GCN

Seit der ersten Erscheinung von GNNs zeigen diese hohe Fähigkeiten bei der effektiven Erfassung struktureller Informationen von Texten, scheitern aber bei der Extraktion von semantischen Informationen von Wörtern während der Komposition auf Textebene (Zhao et al., 2024).

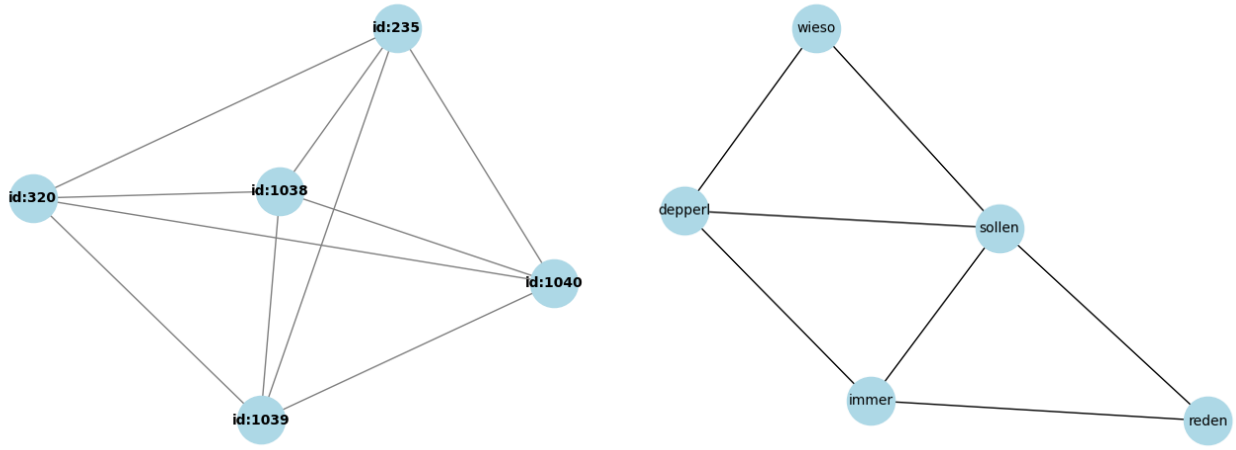

Abb. 4: Konstruktion der Co-Occurrence Graphen nach kompletter (links) und Sliding-Window (rechts) Methode

Um diese Einschränkungen zu umgehen, werden immer mehr hybride Modelle eingesetzt, welche die Stärken von BERT und GNN/GCNs kombinieren. So kann man die Outputs von BERT ähnlich wie beim Ensemble-Learning verstärken. Deren Implementation erfordert jedoch deutlich mehr Aufwand als die vorherigen Modelle. Unsere Architektur folgt einem ähnlichen Prozess wie die aus Abbildung drei. Anstatt einige Schichten hinzuzufügen, muss die Architektur aufgeteilt werden. Jeder Text wird auf zwei unterschiedliche Arten verarbeitet. Einmal wird der Text durch das traditionellen BERT Vorgehen verarbeitet, sprich tokenisiert und anschließend werden von einem BERT-Modell Merkmale auf Grundlage der Embeddings extrahiert. Zur gleichen Zeit erstellen wir auf Basis des Trainingsdatensatzes ein Vokabular. Es wird nur der Trainingsdatensatz verwendet, um Data Leakage zu vermeiden. Dieses Vokabular, nach spezifischer, Graph-orientierter Vorverarbeitung der Texte (Entfernung von Stoppwörter und Normalisierung der Wörter) enthält somit nur noch einzelne Wörter. Für die Vorverarbeitung in deutscher Sprache haben wir die trainierte Pipeline von spaCy *de_core_news_sm* verwendet. Auf Basis des Vokabulars bilden wir die Graphen für jeden Eingabetext aus dem Datensatz. Diese Graphen dienen als Eingabe für die GNNs, um so lokale Zusammenhänge innerhalb der Texte zu detektieren. An dieser Stelle existieren zwei unterschiedliche Möglichkeiten zur Erzeugung der Graphen. Beide Verfahren sind in Abbildung vier dargestellt. Die Graphen beruhen auf dem Hassrede Text “wieso ‘Depperl’ du sollst doch nicht immer von dir reden”. Der linke Graphen beruht auf einer kompletten Co-Occurrence. Diese Art wurde auch im Forschungsartikel aus Abbildung drei verwendet. Hierbei wird jedes Wort bzw. Token mit jedem Wort im Satz verbunden. So entsteht ein Graph, wo jedes Wort miteinander verbunden ist. Der rechte Graph aus Abbildung vier zeigt die Erstellung unter Verwendung eines Sliding-Window Verfahrens. Dabei handelt es sich um eine lokale Co-Occurrence, bei der zwei Wörter im Graphen nur dann miteinander verbunden werden, wenn sie innerhalb eines Fensters der Größe drei im Text erscheinen. Dieses Fenster kann während den Experimenten angepasst werden, was vor allem für längere Sequenzen sinnvoll wäre. Des weiteren haben wir die Graphen einmal mit den entsprechenden Tokens aus dem Vokabular und einmal mit den eigentlichen Wörtern dargestellt. Natürlich wird beim Training die erste Variante verwendet. Die rechte dient in diesem Fall nur der vereinfachten Abbildung der Graphen. Man erkennt ebenfalls, dass das Vokabular keine Stoppwörter mehr enthält. Der gleiche Text nimmt folgende Form in dem BERT Modell an: 'input_ids': tensor ([3, 6724, 26910, 26944, 25268, 608, 26907, 26944, 4547, 459, 13, 1575, 149, 922, 88, 14843, 11614, 4, 0, 0, …]), 'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, …]), wobei die Punkte jeweils das Array mit Nullen auffüllen, bis die Gesamtlänge, in diesem Fall 128, erreicht wurde. Die attention_mask gibt an, welche Tokens vom Modell zu berücksichtigen sind. BERT verarbeitet diese Tokens und nach mehreren Schichten erhalten wir die endgültige Repräsentation der Wortmerkmale, wo wir unter anderem auf das CLS Token zugreifen. Der zweite Teil der Architektur verwendet komplexe GNN Schichten, um anhand der Graphen Merkmale zu extrahieren, welche dann zusammen mit den Outputs von BERT kombiniert werden, um stabilere Resultate zu erzeugen. Neben einfachen Schichten wie GCNConv setzen wir auch GATConv (Graph Attention Network) und SAGEConv (SAmple and aggreGatE) ein. GATConv Schichten nutzen Aufmerksamkeitsmechanismen (engl. attention mechanism), ähnlich wie BERT, um zu lernen, welche Nachbarn eines Knoten, in unserem Fall Text-Embeddings, wichtiger sind. SAGEConv eignet sich vor allem für größere Graph Repräsentationen. Der Ansatz von SAGEConv ist in Abbildung fünf dargestellt. Es wird eine Reihe von Aggregatorfunktionen eingesetzt, die darauf trainiert sind, Merkmalsinformationen aus der lokalen Nachbarschaft eines Knotens zu aggregieren, wobei jede Funktion Informationen aus einer unterschiedlichen Anzahl von Sprüngen (engl. hops) bzw. Suchtiefen um den jeweiligen Knoten herum berücksichtigt (Hamilton et al., 2018). So können gezielt Merkmale über die Wichtigkeit von bestimmten Wörtern und deren Nachbarn festgehalten werden. Die Aggregatorfunktionen bestehen hauptsächlich aus dem Mittelwert oder einzelnen LSTM (Hamilton et al., 2018). SAGEConv eignet sich besonders durch die gelernten Aggregatorfunktionen für die Inferenz auf unbekannten Graphen. Für SAGEConv und GATConv verwenden wir wegen der Wichtigkeit der Nachbarknoten Sliding-Window.

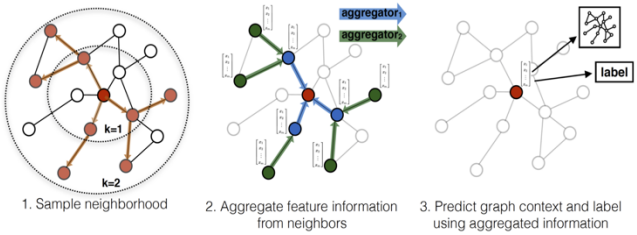

Abb. 5: GraphSAGE Ansatz (Hamilton et al., 2018)

E. LSTM

Wie in Abschnitt *C* bereits erläutert, basieren LSTMs auf RNNs und stellen eine notwendige erweiterte Version der herkömmlichen RNN-Architektur dar. LSTM-Modelle haben die Fähigkeit, beliebige Informationen zu vergessen oder wiederherzustellen (S et al., 2022). In Kombination mit drei Gate-Mechanismen entscheidet das Modell selbst, welche Informationen berücksichtigt werden und welche nicht. Durch ausgewählte Aktivierungsfunktionen werden LSTM-Architekturen oft eingesetzt, um unter anderem das Problem der verschwindenden Gradienten zu vermeiden. LSTMs können relativ einfach, ähnlich wie CNNs, dem Modell hinzugefügt werden. Dabei ist zu bemerken, dass die Angabe *bidirectional=True* wichtig ist, um wie BERT die Texte von links und rechts gleichzeitig zu betrachten. Diese Vorgehensweise ist ähnlich zum Ansatz von Alghamdi et al. (2022). Der genaue Aufbau des BERT-LSTM, aber auch aller anderen Modelle, kann aus dem im Abstrakt angegebenen Github Repository entnommen werden.

VI. Ergebnisse der Experimente

Wir testen die Leistung der einzelnen Erweiterungen der BERT-Architektur auf dem HOCON34k Datensatz wie wir ihn in Kapitel vier definiert haben. Den experimentellen Aufbau skizzieren wir in Abschnitt *A*. In Abschnitt *B* dieses Kapitels präsentieren wir dann die jeweiligen Ergebnisse.

A. Experimenteller Aufbau

Zur Einordnung der empirischen Ergebnisse und um die Modelle untereinander wissenschaftlich vergleichen zu können, verwenden wir eine Handvoll vorgefertigter Metriken. Die zentrale Metrik zum Evaluieren der Modelle bildet der S-Score. Diese wurde aus dem HOCON34k Forschungsartikel entnommen und identisch implementiert (Keller et al., 2025). Der S-Score ist ideal, um machine learning Modelle zur Klassifikation von Hassrede zu evaluieren, da dieser so robust wie möglich und auf den exakten Anwendungsfall zugeschnitten ist (Keller et al., 2025). Der S-Score legt besonderen Wert auf false negatives (FN), da diese, wenn nicht vom Model erkannt, gravierendere Folgen haben als false positives (FP). Die genaue Formel mit den dazugehörigen Metriken haben wir eins zu eins aus der Forschung übernommen und sind in Gleichungen zwei bis vier dargestellt (Keller et al., 2025).

$$F_2 = \frac{(1+2)^2 \times TP}{(1+2)^2 \times TP + FP + 2^2 \times FN} \quad (2)$$

$$M^{norm} = \frac{MCC - (-1)}{1 - (-1)} = \frac{MCC + 1}{2} \quad (3)$$

$$S = \frac{M^{norm} + F_2^{binary}}{2} \quad (4)$$

Die Metrik MCC steht für den Matthews Korrelations Koeffizienten. Für den S-Score wird eine binäre Darstellung der F_2 Metrik verwendet, der S-Score gibt somit Werte im Bereich [0;1] zurück (Keller et al., 2025). Neben diesen Metriken geben wir auch die Recall, Präzision und F_1 Metrik mit an. Das ganze Training findet auf einer NVIDIA GeForce RTX 2080 SUPER mit folgenden Abhängigkeiten statt: PyTorch 2.6.0; NumPy 2.1.1; Matplotlib 3.10.1; Transformers 4.49.0; Python 3.12.5; CUDA 11.8. Obwohl wir kein konkretes Hyperparametertuning vornehmen, verwenden wir dennoch Optuna für das Trainieren der Modelle. Dies ermöglicht das Speichern der besten Modellinstanz innerhalb einer einzelnen Epoche, anstatt nur das Modell am Ende eines Durchlaufs zu berücksichtigen. Da jede Architektur unter den gleichen Bedingungen trainiert werden und die Ergebnisse reproduzierbar sein sollen, haben wir nur jeweils sehr kleine bis keine Bereiche verwendet, in denen sich die jeweiligen Hyperparameter während einer Optuna Studie bewegen. Wir haben uns für folgende Bereiche entschieden: Lernrate =[3e-5,1e-4]; Batch Größe (nicht für GNN/GCN) = [32,64]; Dropout = [0.1,0.5]; Weight Decay = [0.01,0.1]; Epochen = [4,4]. Als Optimierer setzen wir AdamW ein. Die Epochen variieren nicht, sodass jedes Modell pro Versuch (engl. trial) während 4 Epochen trainiert wird. Eine Optuna Studie besteht aus jeweils zehn Versuchen. Jedes Modell minimiert eine Kategorische Kreuzentropie-Verlustfunktion (engl. cross entropy loss) zusammen mit den in Kapitel vier definierten Klassengewichten. Dadurch sind die Architekturen ebenfalls leichter auf multi Klassen Klassifikationen zu skalieren. Das BERT-Modell und der Tokenizer beruhen auf dem *bert-base-german-cased* Modell.

B. Ergebnisse

Die Ergebnisse sind in Tabelle zwei dargestellt. Dabei wurden die Werte der besten Modelle innerhalb der einzelnen Optuna Studien abgebildet. *Bert-FC wurde mit ReLu, Leaky, Sigmoid und tanh trainiert. Wir haben nur das beste Ergebnis beibehalten, nämlich das Modell mit 4 Schichten unter Verwendung der Aktivierungsfunktion LeakyReLu zwischen jeder fully connencted Schicht. Das BERT-LSTM Modell haben wir neben zwei und vier Schichten auch mit Attention trainiert. Das beste Resultat beruht jedoch auf der bidirektionalen Implementation mit zwei Schichten. Die Werte aus der Tabelle der BERT-CNNmulti Architektur wurde bei konstanter Haltung der Output Dimensionen erreicht. Eine zu starke Reduzierung der Dimensionen in jeder Schicht hat die Ergebnisse verschlechtert.

Modell	Präz.	Rec.	F_1	F_2	S
baseline	0.2825	0.7686	0.4132	0.5718	0.6119
BERT-CNNsimple	0.2931	0.8031	0.4294	0.5957	**0.6306**
BERT-CNNmulti	0.2984	0.7725	0.4305	0.5862	0.6249
BERT-FC*	0.2993	0.7667	0.4305	0.5842	0.6237
BERT-RNN	0.2770	0.7935	0.4107	0.5780	0.6151
BERT-GCN	0.2534	0.8604	0.3915	0.5817	0.6134
BERT-SAGEConv	0.2136	<u>0.9216</u>	0.3468	0.5542	0.5858
BERT-GATConv	0.2670	0.8241	0.4034	0.5815	0.6157
BERT-LSTM	0.2944	0.7706	0.4260	0.5822	0.6214

Tabelle 2: Ergebnisse auf dem Validationsdatensatz

Vereinzelt haben wir Modelle mit fixen Parametern trainiert. Dies wurde eingesetzt, um die Metriken innerhalb eines Optuna-Versuchs zu bestätigen. Die BERT-RNN Werte bestehen aus einer Architektur mit zwei RNN Schichten und jeweils einer LeakyReLu Aktivierungsfunktion. Die BERT-GCN Architektur, welche mit dem Wert aus der Tabelle zwei korrespondiert, besteht aus zwei GCN Schichten. Bei den einzelnen Graph-Architekturen wurde jeweils das Sliding-Window Verfahren verwendet. Des weiteren wurde eine Embedding Schicht hinzugefügt, um die Tokens aus dem Vokabular in Embeddings umzuwandeln und so Vektoren mit Kontextinformationen zu erhalten. Die Ergebnisse unter kompletter Co-Occurrence waren deutlich schlechter. Neben

der Tabelle zwei sind die aufgerundeten Recall Werte der einzelnen Erweiterungen in Abbildung sechs dargestellt. Die Trainingsdauer war über alle Modelle hinweg konstant. Das Training der graphisch orientierten Architekturen dauerte am längsten, mit einer durchschnittlichen Laufzeit von neun Minuten. Jedes Modell zeigte ein leichtes bis mittleres Overfitting ab jeweils der zweiten Epoche auf, sodass in einem zweiten Durchlauf die Epochen auf jeweils zwei reduziert wurden.

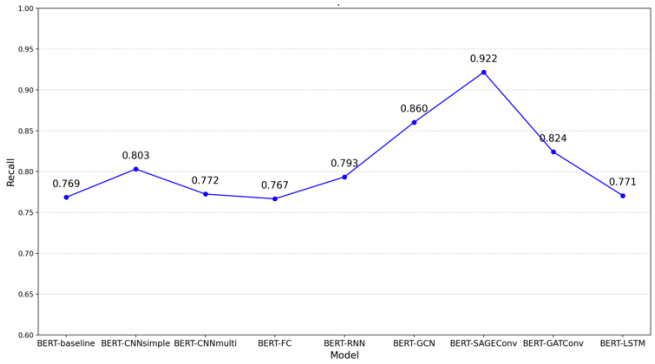

Abb. 6: Recall Werte der Modelle

VII. Evaluation der Modelle

Betrachtet man bloß den S-Score aus Tabelle zwei, dann beruht das beste Modell auf der BERT-CNNsimple Architektur. Das Modell hat den S-Score im Vergleich zum baseline Modell um 1.9% erhöht. Im Kontext der automatisierten Klassifikation von Hassrede und dem verwendeten Datensatz stellt dies bereits eine signifikante Verbesserung dar. Die klassischen Erweiterungen (FC, RNN, LSTM) haben ebenfalls einen höheren S-Score, wenn auch nur minimal höher, als das baseline Modell. Die GCN Modelle haben im Bereich des S-Scores nicht signifikant besser abgeschnitten. Besonders interessant ist jedoch die Betrachtung des Recall Wertes. Diese sind in Abbildung sechs festgehalten. Obwohl BERT-SAGEConv und BERT-GATConv bei den restlichen Werten sehr nah an oder sogar unter dem baseline Modell liegen, erzielen sie extrem hohe Recall Werte. Dies bedeutet, dass die Modelle dank der Einbettung von Graphen besonders präzise in der Klassifikation der positiven Klasse, also der Hassrede Instanzen, sind. Die BERT-SAGEConv Architektur hat sogar einen Recall-Wert von 0.9216, was besonders hoch ist. Dies liegt eventuell an den zusätzlichen Informationen welche durch die zwei SAGEConv Schichten aus den Graphen gewonnen werden. Die SAGEConv Schichten sind des weiteren speziell für Inferenz auf unbekannten Graphen konzipiert, was einen Einsatz in der Realität zur gezielten Detektion von Hassrede ermöglicht. BERT-GCN kann als mittelweg eingesetzt werden, weil sowohl der S-Score als auch der Recall Wert über dem Durchschnitt liegen. Hierfür müsste jedoch noch die Effizienz im Rahmen der Inferenz getestet werden. Um die Ergebnisse zu konkretisieren, haben wir noch eine Konfusionsmatrix in Abbildung sieben am Ende des Dokuments hinzugefügt. Diese zeigt die Ergebnisse des BERT-CNNsimple Modells auf dem Test Datensatz.

VIII. Fazit

Insgesamt deuten die Ergebnisse darauf hin, dass eine Erweiterung der Standard BERT Architektur Sinn macht, wenn man die Leistung bei der automatisierten Klassifikation von Hassrede steigern möchte. Im Allgemeinen haben alle implementierten Modelle das baseline Modell übertroffen, sodass von einer globalen Verbesserung der Klassifikation gesprochen werden kann. Unsere Experimente zeigen, dass die verschiedenen Modelle sich neben ihrer Architektur auch in den erzielten Metriken unterscheiden. Vor allem der Einsatz von CNN Schichten in Kombination mit der BERT-Architektur hat sich als äußerst sinnvoll erwiesen. Unsere Experimente mit Ensemble Learning anhand von Graphen hat ebenfalls spannende Ergebnisse geliefert. Die Modelle sind durch die hohen Recall Werte besonders vielversprechend, wenn es darum geht, keine Hassrede zu übersehen. Da die beiden CNN Erweiterungen die beiden besten S-Scores erzielen, wäre eine Kombination dieser Architekturen mit den Elementen der GCN Ansätzen eventuell von Vorteil. In zukünftigen Forschungen könnten diese Architekturen noch weiter entwickelt werden, sodass man die Vorteile der besten Modelle miteinander kombinieren kann. Unsere Forschung hat ebenfalls gezeigt, wie sensibel die BERT-Architektur auf Veränderungen reagiert. Auch die Wichtigkeit der korrekten Hyperparameter stand im Vordergrund. Allgemein kann man sagen, dass die Manipulation der herkömmlichen BERT Architektur einen Mehrwert hat. Man muss jedoch auch sagen, dass wegen der doch minimalen Unterschiede zwischen den einzelnen Modellen der Fokus an erster Stelle auf dem eingesetzten Datensatz liegen sollte. Die Ergebnisse deuten darauf hin, wie wichtig der Datensatz eigentlich ist, denn egal wie kompliziert das Modell am Ende ist, ohne qualitative Daten werden nur minimale Verbesserungen erzielt. Da unsere Forschung zeitlich begrenzt war, wäre es sinnvoll, die Modelle noch auf anderen Datensätzen und Benchmarks zu evaluieren. Wir haben aus diesem Grund ebenfalls nur das Standard deutsche BERT-Modell verwendet. Zum Beispiel erreicht die gleiche BERT-Architektur mit dem *gbert-large* anstelle von dem *bert-base-german-cased* Modell bereits einen S-Score von 0,638184 ohne irgendwelche Erweiterungen. Vereinzelte Modelle haben gezeigt, dass tiefere Architekturen nicht unbedingt bessere Ergebnisse liefern, wie es bei dem BERT-CNNmulti Modell der Fall ist. Der Fokus sollte dementsprechend laut unseren Erfahrungen an erster Stelle auf dem Datensatz und einfachen Erweiterungen liegen. In Kombination mit den Graphen hatte unsere Implementation ein Problem mit nicht bekannten Tokens, was in zukünftigen Forschungen eventuell behoben werden kann. Zusammenfassend hat unsere Forschung einen klaren als auch strukturierten Einblick in die Möglichkeiten der Erweiterungen der BERT-Architektur ermöglicht. Durch unsere Experimente wurde klar, welche Erweiterungen sich am besten für die Klassifikation von Hassrede eignen.

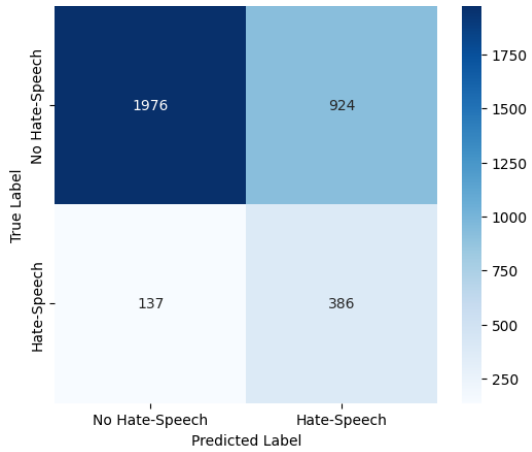

Abb. 7: Konfusionsmatrix BERT-CNNsimple Testdatensatz

IX. Literaturverzeichnis

Acheampong, F., Nunoo-Mensah, H., & Chen, W. (2021). *Transformer Models for Text-based Emotion Detection: A Review of BERT-based Approaches*.

Aftan, S., & Shah, H. (2023). A Survey on BERT and Its Applications. *2023 20th Learning and Technology Conference (L&T)*, 161–166. https://doi.org/10.1109/LT58159.2023.10092289

Alghamdi, J., Lin, Y., & Luo, S. (2022). Modeling Fake News Detection Using BERT-CNN-BiLSTM Architecture. *2022 IEEE 5th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, 354–357. https://doi.org/10.1109/MIPR54900.2022.00069

Alyoubi, K. H., Alotaibi, F. S., Kumar, A., Gupta, V., & Sharma, A. (2023). A novel multi-layer feature fusion-based BERT-CNN for sentence representation learning and classification. *Robotic Intelligence and Automation*, *43*(6), 704–715. https://doi.org/10.1108/RIA-04-2023-0047

Bai, X. (2018). Text classification based on LSTM and attention. *2018 Thirteenth International Conference on Digital Information Management (ICDIM)*, 29–32. https://doi.org/10.1109/ICDIM.2018.8847061

Beniwal, R., & Saraswat, P. (2024). A Hybrid BERT-CNN Approach for Depression Detection on Social Media Using Multimodal Data. *The Computer Journal*, *67*(7), 2453–2472. https://doi.org/10.1093/comjnl/bxae018

Chen, X., Cong, P., & Lv, S. (2022). A Long-Text Classification Method of Chinese News Based on BERT and CNN. *IEEE Access*, *10*, 34046–34057. https://doi.org/10.1109/ACCESS.2022.3162614

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* (No. arXiv:1810.04805; Version 2). arXiv. https://doi.org/10.48550/arXiv.1810.04805

Dong, J., He, F., Guo, Y., & Zhang, H. (2020). A Commodity Review Sentiment Analysis Based on BERT-CNN Model. *2020 5th International Conference on Computer and Communication Systems (ICCCS)*, 143–147. https://doi.org/10.1109/ICCCS49078.2020.9118434

Eang, C., & Lee, S. (2024). Improving the Accuracy and Effectiveness of Text Classification Based on the Integration of the Bert Model and a Recurrent Neural Network (RNN_Bert_Based). *Applied Sciences*, *14*(18), Article 18. https://doi.org/10.3390/app14188388

Gao, Z., Feng, A., Song, X., & Wu, X. (2019). Target-Dependent Sentiment Classification With BERT. *IEEE Access*, *7*, 154290–154299. https://doi.org/10.1109/ACCESS.2019.2946594

Hamilton, W. L., Ying, R., & Leskovec, J. (2018). *Inductive Representation Learning on Large Graphs* (No. arXiv:1706.02216). arXiv. https://doi.org/10.48550/arXiv.1706.02216

Huang, Y.-H., Chen, Y.-H., & Chen, Y.-S. (2022). ConTextING: Granting Document-Wise Contextual Embeddings to Graph Neural Networks for Inductive Text Classification. In N. Calzolari, C.-R. Huang, H. Kim, J. Pustejovsky, L. Wanner, K.-S. Choi, P.-M. Ryu, H.-H. Chen, L. Donatelli, H. Ji, S. Kurohashi, P. Paggio, N. Xue, S. Kim, Y. Hahm, Z. He, T. K. Lee, E. Santus, F. Bond, & S.-H. Na (Hrsg.), *Proceedings of the 29th International Conference on Computational Linguistics* (S. 1163–1168). International Committee on Computational Linguistics. https://aclanthology.org/2022.coling-1.100/

Jin, Y., & Zhao, A. (2024). Bert-based graph unlinked embedding for sentiment analysis. *Complex & Intelligent Systems*, *10*(2), 2627–2638. https://doi.org/10.1007/s40747-023-01289-9

Kaur, K., & Kaur, P. (2023). BERT-CNN: Improving BERT for Requirements Classification using CNN. *Procedia Computer Science*, *218*, 2604–2611. https://doi.org/10.1016/j.procs.2023.01.234

Keller, M.-E., Auch, M., Döschl, A., Vlk, F., Quernheim, J., Hartmann, M., Mandl, P., Kaul, A., & Franz, M. (2025). HOCON34k: A Corpus of Hate Speech in Online Comments from German Newspapers. In P. Delir Haghighi, M. Greguš, G. Kotsis, & I. Khalil (Hrsg.), *Information Integration and Web Intelligence* (S. 212–226). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-78090-5_18

Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., El Emam, K., & Rosenberg, J. (2002). Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, *28*(8), 721–734. IEEE Transactions on Software Engineering. https://doi.org/10.1109/TSE.2002.1027796

Kovaleva, O., Romanov, A., Rogers, A., & Rumshisky, A. (2019). *Revealing the Dark Secrets of BERT* (No. arXiv:1908.08593). arXiv. https://doi.org/10.48550/arXiv.1908.08593

Liu, H., & Xie, L. (2021). Research on Sarcasm Detection of News Headlines Based on Bert-LSTM. *2021 IEEE International Conference on Emergency Science and Information Technology (ICESIT)*, 89–92. https://doi.org/10.1109/ICESIT53460.2021.9696851

Liu, P., Wang, X., Wang, L., Ye, W., Xi, X., & Zhang, S. (2021). Distilling Knowledge from BERT into Simple Fully Connected Neural Networks for Efficient Vertical Retrieval. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 3965–3975. https://doi.org/10.1145/3459637.3481909

Lu, Z., Du, P., & Nie, J.-Y. (2020). VGCN-BERT: Augmenting BERT with Graph Embedding for Text Classification. *Advances in Information Retrieval*, *12035*, 369–382. https://doi.org/10.1007/978-3-030-45439-5_25

Muralitharan, J., & Arumugam, C. (2024). Privacy BERT-LSTM: A novel NLP algorithm for sensitive information detection in textual documents. *Neural Computing and Applications*, *36*(25), 15439–15454. https://doi.org/10.1007/s00521-024-09707-w

Pandey, R., & Singh, J. P. (2023). BERT-LSTM model for sarcasm detection in code-mixed social media post. *Journal of Intelligent Information Systems*, *60*(1), 235–254. https://doi.org/10.1007/s10844-022-00755-z

PBV, R. R., Prasad, M., Pokkuluri, K. S., Srikanth, P., Dangeti, S. R., & Rao, B. V. (2025). An Efficient Sentiment Classification Model Using Fusion of BERT and Deep Learning RNN Variants. In P. Pareek, S. Mishra, M. J. C. S. Reis, & N. Gupta (Hrsg.), *Cognitive Computing and Cyber Physical Systems* (S. 268–278). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-77075-3_22

Perin, E. L. S., Souza, M. C. de, Silva, J. de A., & Matsubara, E. T. (2025). DynGraph-BERT: Combining BERT and GNN Using Dynamic Graphs for Inductive Semi-Supervised Text Classification. *Informatics*, *12*(1), Article 1. https://doi.org/10.3390/informatics12010020

Putra, C. D., & Wang, H.-C. (2024). Advanced BERT-CNN for Hate Speech Detection. *Procedia Computer Science*, *234*, 239–246. https://doi.org/10.1016/j.procs.2024.02.170

R. Abas, A., Elhenawy, I., Zidan, M., & Othman, M. (2022). BERT-CNN: A Deep Learning Model for Detecting Emotions from Text. *Computers, Materials & Continua*, *71*(2), 2943–2961. https://doi.org/10.32604/cmc.2022.021671

Rodrigues Makiuchi, M., Warnita, T., Uto, K., & Shinoda, K. (2019). Multimodal Fusion of BERT-CNN and Gated CNN Representations for Depression Detection. *Proceedings of the 9th International on Audio/Visual Emotion Challenge and Workshop*, 55–63. https://doi.org/10.1145/3347320.3357694

Roethel, A., Ganzha, M., & Wróblewska, A. (2023). *Enriching language models with graph-based context information to better understand textual data* (No. arXiv:2305.11070). arXiv. https://doi.org/10.48550/arXiv.2305.11070

S, S., Sunagar, P., Rajarajeswari, S., & Kanavalli, A. (2022). BERT-Based Hybrid RNN Model for Multi-class Text Classification to Study the Effect of Pre-trained Word Embeddings. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *13*(9), Article 9. https://doi.org/10.14569/IJACSA.2022.0130979

Safaya, A., Abdullatif, M., & Yuret, D. (2020). *KUISAIL at SemEval-2020 Task 12: BERT-CNN for Offensive Speech Identification in Social Media* (No. arXiv:2007.13184). arXiv. https://doi.org/10.48550/arXiv.2007.13184

Saraswat, P., & Beniwal, R. (2024). BERT-based RNN for Effective Detection of Depression with Severity Levels from Text Data. *2024 IEEE Symposium on Wireless Technology & Applications (ISWTA)*, 52–56. https://doi.org/10.1109/ISWTA62130.2024.10651873

Shah, K., Patel, H., Sanghvi, D., & Shah, M. (2020). A Comparative Analysis of Logistic Regression, Random Forest and KNN Models for the Text Classification. *Augmented Human Research*, *5*(1), 12. https://doi.org/10.1007/s41133-020-00032-0

Sun, C., Qiu, X., Xu, Y., & Huang, X. (2020). *How to Fine-Tune BERT for Text Classification?* (No. arXiv:1905.05583). arXiv. https://doi.org/10.48550/arXiv.1905.05583

Taggu, A., & Teyi, N. (2025). Improving Twitter Sentiment Analysis with a Hybrid BERT and Graph Neural Network Ensemble. In Z. Mohamed, F. Hassan, G. Tan, A. Ahmad, L. Pei Ling, & S. Buyamin (Hrsg.), *Systems Modelling and Simulation* (S. 159–170). Springer Nature. https://doi.org/10.1007/978-981-96-4613-5_12

Wan, C.-X., & Li, B. (2022). Financial causal sentence recognition based on BERT-CNN text classification. *The Journal of Supercomputing*, *78*(5), 6503–6527. https://doi.org/10.1007/s11227-021-04097-5

Xin, C., & Zakaria, L. Q. (2024). Integrating Bert With CNN and BiLSTM for Explainable Detection of Depression in Social Media Contents. *IEEE Access*, *12*, 161203–161212. https://doi.org/10.1109/ACCESS.2024.3488081

Yang, Y., & Cui, X. (2021). Bert-Enhanced Text Graph Neural Network for Classification. *Entropy*, *23*(11), Article 11. https://doi.org/10.3390/e23111536

Zhao, J., Qin, X., Peng, L., Zhang, J., Wang, W., & Zheng, H. (2024). BGNN-TC: BERT GNN Based Text Classification Algorithm. *2024 IEEE First International Conference on Data Intelligence and Innovative Application (DIIA)*, 1–8. https://doi.org/10.1109/DIIA62678.2024.10871385

Zheng, S., & Yang, M. (2019). A New Method of Improving BERT for Text Classification. In Z. Cui, J. Pan, S. Zhang, L. Xiao, & J. Yang (Hrsg.), *Intelligence Science and Big Data Engineering. Big Data and Machine Learning* (S. 442–452). Springer International Publishing. https://doi.org/10.1007/978-3-030-36204-1_37

Zhou, C., Sun, C., Liu, Z., & Lau, F. C. M. (2015). *A C-LSTM Neural Network for Text Classification* (No. arXiv:1511.08630). arXiv. https://doi.org/10.48550/arXiv.1511.08630