HAW HAMBURG

BACHELOR THESIS
Qiuzhu Lu

Textbasierte Emotionserkennung durch neuronale Netze

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Qiuzhu Lu

Textbasierte Emotionserkennung durch neuronale Netze

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Angewandte Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Michael Neitzke
Zweitgutachter: Prof. Dr. Peer Stelldinger

Eingereicht am: 05. Oktober 2023

Qiuzhu Lu

Thema der Arbeit

Textbasierte Emotionserkennung durch neuronale Netze

Stichworte

Künstliche Intelligenz, Maschinelles Lernen, Neuronale Netze, Natürliche Sprachverarbeitung, Emotionserkennung, CNN, LSTM, LSTM-CNN, CNN-LSTM, Ensemble, Erklärbare KI, LIME, Spatial Dropout

Kurzzusammenfassung

In der Bachelorarbeit werden verschiedene Architekturen neuronaler Netze untersucht, um Emotionen aus Texten zu erkennen, was ein Teil der natürlichen Sprachverarbeitung ist. Ziel ist es, das leistungsfähigste Netz zu finden und zu überprüfen, ob es Texte gemäß der bedeutungsvollen Wörtern klassifiziert.

LSTM-, CNN-, LSTM-CNN-, CNN-LSTM- und Ensemble-Modelle werden anhand des englischsprachigen GoEmotions-Datensatzes trainiert und verglichen. Für jeden Modelltyp werden drei verschiedene Worteinbettungen untersucht, nämlich GloVe-Tweets, GloVe-6B und Word2Vec. Darüber hinaus werden die Einflüsse der Anzahl der LSTM-Units und einer Spatial-Dropout-Schicht auf die Modelle untersucht. Das beste Modell ist das LSTM-CNN-Modell mit der GloVe-Tweets-Worteinbettung, das eine Genauigkeit von 62,61% und einen F1-Score von 57,14% aufweist. Die Klassifizierungen des LSTM-CNN-Modells werden mit LIME erklärt. Nicht alle Wörtern, die zur Klassifizierung führen, sind sinnvoll. Darüber hinaus haben die anderen Modelle mit der Word2Vec-Worteinbettung die beste Leistung erzielt. Die Verwendung einer Spatial-Dropout-Schicht verbessert die Leistung des CNN-Modells. Allerdings verbessert es nicht immer die Leistungen anderer Modelle. Es zeigt sich zudem, dass eine Verdoppelung der LSTM-Units keine bessere Leistung garantiert.

Qiuzhu Lu

Title of Thesis

Text-based emotion recognition using neural networks

Keywords

Artificial intelligence, Machine learning, Neural networks, Natural language processing, Emotion recognition, CNN, LSTM, LSTM-CNN, CNN-LSTM, Ensemble, Explained AI, LIME, Spatial dropout

Abstract

This bachelor thesis deals with the recognition of emotions from texts using different architectures of neural networks. Emotion recognition belongs to a subfield of natural language processing. The goal is to find the best performing network and verify that it classifies texts according to meaningful words.

LSTM, CNN, LSTM-CNN, CNN-LSTM and Ensemble models are used and compared using the English dataset GoEmotions. For each model type, three different word embeddings are exprimented, namely GloVe-Tweets, GloVe-6B, and Word2Vec. In addition, the influences of the number of LSTM units and a spatial dropout layer on the models are researched. The best model is LSTM-CNN model with the GloVe-Tweets word embedding, which has an accuracy of 62.61% and a F1 score of 57.14%. The classifications of the LSTM-CNN model are explained using LIME. Not all words leading to the classification are meaningful. In addition, the other models with Word2Vec word embedding have achieved the best performance. Moreover, using a spatial dropout layer improves the performance of CNN model. However, it does not always improve the performance of the other models. It also shows that doubling the LSTM units doesn't guarantee better performance.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

Adam Adaptive Moment Estimation.

BILSTM bidirectional LSTM.

CBOW Continuous Bag of Words.

CNN Convolutional neural network.

KI Künstliche Intelligenz.

LIME Local Interpretable Model-agnostic Explanations.

LSTM Long Short-Term Memory.

SP-LIME Submodular Pick Local Interpretable Model-agnostic Explanations.

Val-Datensatz Validation-Datensatz.

1 Einleitung

1.1 Hintergrund

Emotionen spielen im Alltag eine wichtige Rolle und beeinflussen Entscheidungsfindungen, soziale Beziehungen und Verhalten. Es gibt immer mehr gefühlsbetonte Textinhalte, wie Microblog-Beiträge und Forumsdiskussionen. Der Bedarf an Analysen der Emotionen in diesen Texten wächst ebenfalls [65]. Zudem werden immer öfter interaktive Dialogsystemen eingesetzt. Die Erkennung von Emotionen trägt dazu bei, die Kommunikation zwischen Mensch und Computer zu verbessern und mentale Zustände zu erkennen [46]. Die Erkennung der Emotion wird in vielen Anwendungen wie der Erkennung der Abschiedsbriefen, der Erkennung beleidigender Texte [9] und im intelligenten Chatbot-Modell [14] verwendet. Die Erkennung von Emotionen in Texten ist derzeit aufgrund ihres großen Anwendungspotenzials in den Bereichen der künstlichen Intelligenz und der Mensch-Computer-Interaktion sehr gefragt [54].

Viele Ansätze der künstlichen Intelligenz sind sehr leistungsfähig. Jedoch funktionieren diese Ansätze nicht transparent. Zum Beispiel werden Wörter auf Vektoren abgebildet, wodurch sie für Menschen nicht mehr verständlich sind [31]. Außerdem sind viele Algorithmen des maschinellen Lernens schwer verständlich, beispielsweise Deep-Learning-basierte Ansätze wie Long Short-Term Memory (LSTM) und Convolutional neural network (CNN). Die Gründe für Entscheidungen eines Künstliche Intelligenz (KI)-Systems werden nur selten erforscht. Die Unverständlichkeit der Ansätze führt dazu, dass die Benutzer weniger Vertrauen in das trainierte System und dessen Entscheidungen haben [47]. Es gibt bereits viele leistungsfähige Modelle der textbasierten Emotionserkennung, deren Genauigkeit bei über 90% liegt. Viele funktionieren aber auch nicht transparent [41]. Um Vertrauen zu schaffen, ist es sehr wichtig, die Entscheidungen eines KI-Systems zu überprüfen. Erklärungen zu den Entscheidungen eines KI-Systems liefern zusätzliche Informationen und helfen den Menschen, das System zu verstehen. Darüber hinaus können die Menschen aufgrund der Erklärungen eventuell bessere Entscheidungen treffen

beziehungsweise sich nicht auf das System verlassen, wenn die Erklärungen keinen Sinn ergeben. Deshalb spielen Erklärungen eine wichtige Rolle bei der Interaktion zwischen Menschen und Maschinen [51]. Mit Ansätzen der erklärbaren künstlichen Intelligenz wird es ermöglicht, Entscheidungen eines Modells zu erklären, beispielsweise mit Local Interpretable Model-agnostic Explanations (LIME) [49].

1.2 Zielsetzung

Ziel der Arbeit ist es, die folgende Frage zu beantworten:

- Mit dem LSTM-Modell, dem CNN-Modell, dem LSTM-CNN-Modell und dem CNN-LSTM-Modell und dem CNN-LSTM-Ensemble werden die Emotionen der Texte klassifiziert. Dann werden diese trainierten KI-Modellen verglichen, welches Modell einen höheren F1-Score und eine höhere Genauigkeit aufweist.
- Mittels LIME werden die Klassifizierungen des besten trainierten Modells erklärt. Auf der Grundlage welcher Wörter hat dieses Modell Entscheidungen getroffen? Es wird bewertet, wie aussagekräftig diese Erklärungen zu den Klassifizierungen sind.

1.3 Aufbau der Arbeit

Kapitel 2 enthält die Grundlagen der neuronalen Netze und der erklärbaren KI. In Kapitel 3 wird die Theorie beschrieben, wie Texte vor dem Training verarbeitet werden. Kapitel 4 gibt einen Überblick über die verwendeten Daten und beschreibt deren Aufbereitung und Einbettung. In Kapitel 5 werden die Experimente verschiedener Modelle und ihre Ergebnisse beschrieben. Es gibt 5 Arten von Modellen: LSTM, CNN, LSTM-CNN, CNN-LSTM und Ensemble. Jeder Typ hat wiederum mehrere Varianten. Darüber hinaus wird jeder Typ unter seinen eigenen Varianten verglichen. Zudem wird das beste Modell jedes Typ ausgewählt. Diese ausgewählte Modelle werden dann in Kapitel 6 miteinander verglichen. Das beste Modell wird mit LIME erklärt und dabei überprüft, ob es die Texte aufgrund der richtigen Wörter klassifizieren kann. Eine Zusammenfassung und ein Ausblick auf zukünftige Arbeiten werden in Kapitel 7 gegeben.

1.4 Verwandte Arbeiten

Diese Arbeit behandelt eine Klassifikationsaufgabe. Darüber hinaus gehört diese Arbeit in den Bereich der Sentiment-Analyse bzw. Emotionsanalyse. In diesem Bereich sind Deep-Learning-basierte Ansätze wichtige Methoden. Shrivastava et al. [56] entwickelten ein sequenzbasiertes CNN mit Worteinbettung zur Erkennung der Emotionen. Im Modell wird eine Attention-Schicht verwendet, sodass sich das Modell bei der Klassifizierung auf die einflussreichen Wörter konzentriert. Pasupa und Seneewong Na Ayutthaya [45] verglichen CNN-, LSTM- und bidirectional LSTM (BILSTM)-Modelle zur Stimmungsanalyse thailändischer Kindergeschichten. Das CNN-Modell weist mit einem F1-Score von 81,70% die besten Werte auf. Zhou und Wu [69] entwickelten ein BiLSTM-Modell und ein LSTM-Modell, um Emotionen der Tweets in 6 Klassen zu klassifizieren. Darüber hinaus kommen in den beiden Modellen Attention-Mechanismen zum Einsatz. In der Arbeite „Emotion Detection using Deep Learning“ [55] werden CNN- und BiLSTM-Modelle mit verschiedenen Methoden der Features-Repräsentationen verglichen. Die Ergebnisse zeigen, dass Features-Repräsentationsmethoden, Textvorverarbeitung und die Anzahl der klassifizierten Emotionsklassen die Leistung der Modelle beeinflussen.

Kim [35] trainierte ein CNN-Model aufbauend auf Word2Vec zur Textklassifizierung. Das Model verwendet mehrere Filter mit unterschiedlichen Fenstergrößen, um mehrere Features zu extrahieren. Anschließend wird eine Max-Over-Time Pooling Operation auf diese Feature-Maps durchgeführt, sodass der maximale Wert für jede Feature-Map ausgewählt wird. Die Ergebnisse beweisen, dass vortrainierte Wortvektoren für verschiedene Klassifizierungsaufgaben verwendet werden können. Ibrahim und Zhang [11] verbesserten das CNN-Modell von Kim, indem eine Spatial-Dropout-Schicht auf Einbettung der Eingabevektoren hinzugefügt wird. Das Ziel des Dropouts besteht darin, Overfitting zu vermeiden.

Saurav et al. [34] entwickelten ein LSTM-CNN-Modell und ein CNN-LSTM-Modell, mit denen Texte nach Emotionen klassifiziert werden. Die Genauigkeit der beiden Modelle liegt jeweils bei 93.88% und 93.20%. Das LSTM-CNN-Modell hat eine bessere Performanz. Iyer et al. [32] trainierten ein LSTM-Modell, ein CNN-Modell und ein CNN-LSTM-Modell, die zur Entwicklung eines Ensembles verwendet werden, das in der Lage ist, menschliche Emotionen mithilfe von Elektroenzephalogramm-Signalen zu erkennen. Es hat eine Genauigkeit von 97,16% erreicht.

Diese Recherche zeigt, dass das CNN-Modell, das LSTM-Modell und ihre kombiniertes Modelle gute Lösungen für die Emotionsanalyse sein können.

2 Grundlagen

2.1 Neuronale Netze

Neuronale Netze sind Teil des maschinellen Lernens und bilden den Kern des Deep-Learnings. Diese Netze simulieren die Funktionsweise der Neuronen im menschlichen Gehirn. Sie bestehen aus einer Eingabeschicht, einer Ausgabeschicht und in den meisten Fällen einer oder mehreren verborgenen Schichten. Die Neuronen sind miteinander verbunden und diese Verbindungen dazwischen haben Gewichtungen. Wenn die Ausgabe eines Neurons über dem Schwellenwert liegt, wird das Neuron aktiviert und diese Daten werden an das nächste Neuron weitergeleitet. [7, 8]

Jedes Neuron hat sein eigenes lineares Regressionsmodell, Eingaben x_i, Gewichtungen w_i, ein Bias b und eine Ausgabe $f(z)$, wobei f eine Aktivierungsfunktion ist. Gewichtungen entscheiden über die Wichtigkeit der Variablen. Die Ausgabe wird nach der folgenden Formel 2.1 berechnet. [7]

$$f(z) = f(\sum_i w_i x_i + b) \tag{2.1}$$

Die Aktivierungsfunktion fügt dem neuronalen Netz Nichtlinearität hinzu. Gäbe es keine Aktivierungsfunktion, würden die Ausgaben nur anhand der Gewichtungen und Bias berechnet werden. Da die Zusammensetzung zweier oder mehrerer linearen Funktionen ebenfalls eine lineare Funktion ist, würde in diesem Fall die Ausgabe eines neuronalen Netzes unabhängig von der Anzahl der verborgenen Schichten wie durch eine lineare Funktion berechnet werden. Die am häufigsten verwendeten Aktivierungsfunktionen sind die Sigmoid-Funktion (2.2), die Tanh-Funktion (2.3) und die ReLU-Funktion (2.4). [13]

$$f(x) = \frac{1}{1 + e^{-x}} \quad , \quad f(x) \in (0, 1) \tag{2.2}$$

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad , \quad f(x) \in (-\infty, +\infty) \tag{2.3}$$

$$f(x) = max(0, x) \quad , \quad f(x) \in [0, +\infty) \tag{2.4}$$

2.1.1 Long Short-Term Memory Netze

LSTM-Netze [30] sind eine besondere Art von RNN-Netze, die 1997 von Sepp Hochreiter und Jürgen Schmidhuber vorgestellt wurden. LSTM-Netze können das Problem des verschwindenden Gradienten von RNN-Netzen lösen. Der Grund dafür ist, dass LSTM-Netze drei Gates besitzen, nämlich das Input-Gate, das Forget-Gate und das Output-Gate. Diese Gates bieten eine bessere Kontrolle über den Gradientenfluss und erhalten besser Langzeitabhängigkeiten [63].

Alle RNN-Netze haben die Form einer Kette von sich wiederholenden Modulen eines neuronalen Netzes. Das repetitive Modul eines RNN-Netz besteht aus nur einer Schicht, während das repetitive Modul eines LSTM-Netz aus vier miteinander interagierten Schichten besteht. [43]

Im Vergleich zu den einfachen RNN-Netzen speichern LSTM-Netze Informationen auch im Zellzustand, der mit wenigen Operationen über die gesamten Module läuft. Folglich können die Informationen im Zellzustand einfach unverändert bleiben. LSTM-Netze können jedoch mithilfe von Gates darüber entscheiden, welche Informationen gespeichert, gelöscht oder weitergeleitet werden. Gates sind Schichten mit Sigmoidfunktion. Da die Ausgabewerte der Sigmoidfunktion zwischen 0 und 1 liegen, können Gates entscheiden, wie viele Informationen weitergeleitet werden. Die folgende Abbildung 2.1 stellt die Schichten, Gates und den Datenfluss des LSTM-Moduls dar. [43]

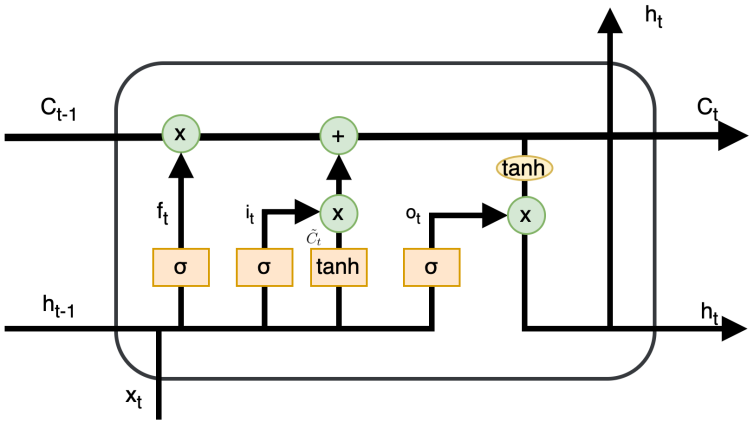


Abbildung 2.1: Repetitives Modul eines LSTM

Forget-Gate-Schicht

Die Forget-Gate-Schicht ist die erste Schicht eines LSTM-Moduls, mit der die LSTM-Netze entscheiden, welche Informationen aus dem Zellzustand gelöscht werden sollen. Die Ausgabe der Forget-Gate-Schicht wird nach der folgenden Formel 2.5 berechnet. [43]

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{2.5}$$

Input-Gate-Schicht

Die Input-Gate-Schicht ist die nächste Schicht mit der Sigmoidfunktion, die entscheidet, welche Werte aktualisiert werden. Die Ausgabe der Input-Gate-Schicht wird nach der folgenden Formel 2.6 berechnet. Hinter der Input-Gate-Schicht befindet sich eine Schicht mit der Tanh-Funktion. Sie erzeugt einen Vektor mit neuen Kandidatenwerten, der nach der Gleichung 2.7 berechnet wird. [43]

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{2.6}$$

$$\tilde{C_t} = tanh(W_C.[h_{t-1}, x_t] + b_C) \tag{2.7}$$

Aktualisierung des Zellenzustands

Dann wird der alte Zellzustand C_{t-1} aktualisiert. Zunächst wird der alte Zellzustand C_{t-1} mit f_t multipliziert, sodass die Werte gelöscht werden können, die vorher durch das Forget-Gate zum Vergessen bestimmt wurden. Dann werden die Kandidatenwerte $\tilde{C_t}$ aus der Input-Gate-Schicht mit i_t multipliziert. Dieser Schritt skaliert die Kandidatenwerten und beeinflusst, wie stark jeder Zustandswert aktualisiert wird. Der neue Wert C_t des Zellzustands ist die Summe der beiden multiplizierten Ergebniswerte, dargestellt durch die Gleichung 2.8. [43]

$$C_t = f_t * C_{t-1} + i_t * \tilde{C_t} \tag{2.8}$$

Output-Gate-Schicht

Die Output-Gate-Schicht entscheidet, welche Werte des Zellzustands ausgegeben werden. Die Ausgabe der Output-Gate-Schicht o_t wird gemäß der Gleichung 2.9 berechnet. Schlussendlich werden die Zellzustandswerte in die Tanh-Funktion eingeben. Die Ausgaben werden mit o_t multipliziert. Wie in der Gleichung 2.10 dargestellt, ist das multiplizierte Ergebnis der Zellzustandsausgabewerte h_t. [43]

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_0) \tag{2.9}$$

$$h_t = o_t * tanh(C_t) \tag{2.10}$$

2.1.2 Convolutional Neural Networks

CNN eignen sich für die Erkennung der Grafiken, die Verarbeitung natürlicher Sprache und die Klassifizierung [12]. Dabei können sie eindimensionale, zweidimensionale und dreidimensionale Daten verarbeiten [17]. Darüber hinaus bestehen sie aus 3 Schichttypen, nämlich der Convolutional-Schicht, der Pooling-Schicht und der vollständig vernetzten Schicht [6]. Die Convolutional-Schicht ist die erste Schicht von CNN, während die vollständige vernetzte Schicht die letzte Schicht von CNN ist.

Convolutional-Schicht
Die Convolutional-Schicht ist der Kern von CNN. Die Convolution ist eine lineare Operation, bei der eine Eingabe mit einer Menge von Gewichtungen multipliziert wird. Die Gewichtungsmatrix bei der Multiplikation wird als Filter oder Kernel bezeichnet. Diese Multiplikation ist jeweils eine elementare Multiplikation zwischen dem Filter und der Eingabe der Filtergröße, nämlich ein Skalarprodukt. Angenommen, dass ein Kernel K ein zwei-dimensionales Feld der Größe $m*n$ ist und eine Eingabe I ebenfalls zwei-dimensional ist. Die zweidimensionale Ausgabe S wird Feature-Map bezeichnet. Die Ausgabe an der Stelle (i, j) wird nach folgender Formel 2.11 berechnet. Häufig werden in einem Netz mehrere Kernel verwendet, damit mehrerer Feature aus der Eingabe extrahiert werden. [17, 26]

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \tag{2.11}$$

Pooling-Schicht
Eine Pooling-Schicht liegt oft hinter einer Convolutional-Schicht. Ihr Ziel ist es, die Anzahl der Eingabeparameter und die Komplexität zu reduzieren. Sie besitzt ebenfalls einen Filter wie eine Convolutional-Schicht, jedoch hat dieser Filter keine Gewichtungen. Eine Aggregationsfunktion wird auf die Werte im Empfangsfeld des Filters angewendet. So wird das Ausgabearray mit den Ergebnissen der Aggregationsfunktion gefüllt. Die am häufigsten verwendeten Aggregationsfunktionen sind das Max-Pooling und das Average-Pooling. Beim Max-Pooling wird der Maximalwert aus dem vom Filter abgedeckten Bereich der Feature-Map ausgewählt, während beim Average-Pooling der Durchschnittswert daraus berechnet wird. [12, 28, 53]

Vollständig vernetzte Schicht
Eine vollständig vernetzte Schicht ist die letzte Schicht des CNN. Jedes Neuron in der Eingabeschicht ist mit einer Aktivierungseinheit oder allen Neuronen in der nächsten Schicht verbunden. Mit dieser Schicht kann die Klassifizierung auf der Grundlage der von Filtern extrahierten Merkmale durchgeführt werden. [28]

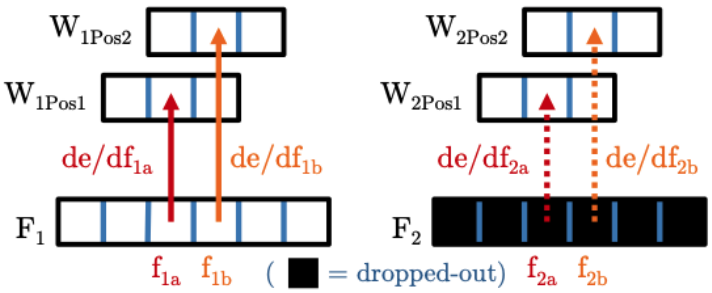
2.1.3 Netzwerktraining

Es gibt zwei grundlegende Ansätze für das Netzwerktraining beim maschinellen Lernen: überwachtes Lernen und unüberwachtes Lernen. Beim überwachten Lernen werden Datensätze mit Labels verwendet. Für Klassifikation- oder Regressionsprobleme wird in der Regel der Ansatz des überwachten Lernens eingesetzt. Aufgrund der Labels können Modelle ihre Genauigkeit messen und dadurch lernen. Beim unüberwachten Lernen werden Datensätze ohne Labels analysiert, um Datenmuster zu erkennen, ohne dass ein menschliches Eingreifen erforderlich ist. [21]

Aufteilung der Datensätze
Vor dem Netzwerktraining werden die Datensätze in 3 Teile geteilt, nämlich Training-Datensatz, Validation-Datensatz (Val-Datensatz) und Test-Datensatz. Der Traning-Datensatz ist der Teil der Daten, der zum Trainieren eines Modells verwendet wird. Ein Modell lernt aus diesen Daten und optimiert seine Parameter anhand dieser Daten. Der Val-Datensatz ist ein Subdatensatz, der zur Änderung der Hyperparameters wie Lernrate verwendet wird. Mit diesen Daten wird die Leistung eines Modells gemessen. Daher helfen diese Daten bei der Auswahl des besten Modells. Der Test-Datensatz ist der Datensatz, der zur Evaluierung eines Modells nach dem Training verwendet wird. Das Ziel der Aufteilung der Datensätze beseht darin, ein Overfitting zu vermeiden. Das tritt auf, wenn ein Modell zu viel an den Training-Datensatz angepasst wird. Infolgedessen passt sich das Modell nur an die Trainingsdaten sehr gut an, jedoch nicht an andere Daten. [24, 64]

Dropout
Eine Maßnahme gegen Overfitting ist Dropout. Dies bezieht sich auf das Ausscheiden von Knoten in einem neuronalen Netz. Alle Verbindungen zwischen den ausgefallenen Knoten und den anderen Konten werden vorübergehend entfernt, sodass ein neues Netz erzeugt wird. Jeder Knoten hat eine Wahrscheinlichkeit auszufallen. Dropout kann sowohl in den verborgenen Schichten als auch in den Eingabeschichten verwendet werden. Zudem benötigt ein Model mit Dropout mehr Epochen zum Trainieren. [67]

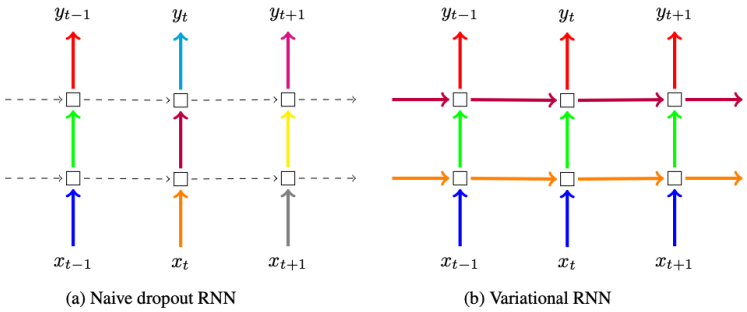
Es gibt verschiedene Arten von Dropout. Spatial-Dropout ist eine Art von Dropout für CNNs, bei dem ganze Feature-Maps aus der Convolutional-Schicht nicht verwendet werden. Dieser Dropout wird in der Abbildung 2.2 dargestellt. Die Kernels für Feature-Maps F_1 und F_2 werden durch die oberen beiden Pixelreihen repräsentiert. Zudem stellen die untere Reihen die ausgegebenen Feature-Maps aus der vorherigen Schicht dar. de/df_2a und de/f_2b sind für den mittleren Pixel des Kernels W_2 während der Backpropagation die Gradientenbeiträge. Da f_2a und f_2b Werte aus derselben Feature-Map sind, korrelieren sie stark. Wenn f_2a oder f_2b herausgenommen werden, bleibt der andere stark korrelierte Gradient weiterhin bestehen. Beim Spatial-Dropout wird die ganze Feature-Map F_2 nicht verwendet und diese stark korrelierende Werte werden daher nicht verwendet. [62]



Quelle: [62]

Abbildung 2.2: Spatial-Dropout nach einer 1D-CNN-Schicht

Ein weiterer Dropout ist der Variational-RNN-Dropout. In der Abbildung 2.3 werden der Naive-RNN-Dropout und der Variational-RNN-Dropout verglichen. Jedes Quadrat steht für eine RNN-Einheit, wobei die horizontalen Pfeile die rekurrente Verbindungen darstellen. Vertikale Pfeile stellen die Eingabe und Ausgabe jeder RNN-Einheit dar. Unterschiedliche Farben entsprechen den verschiedenen Dropout-Masken. Beim Variational-RNN-Dropout wird bei jedem Zeitschritt dieselbe Dropout-Maske verwendet, auch bei den rekurrenten Schichten. Das bedeutet, dass bei jedem Zeitschritt für die Eingabeschichten, Ausgabeschichten und rekurrenten Schichten dieselben Knoten aus dem Netzwerk herausfallen. [23]



Quelle: [23]

Abbildung 2.3: Naive-RNN-Dropout vs Variational-RNN-Dropout

Early Stopping

Darüber hinaus ist auch Early Stopping eine Maßnahme gegen Overfitting. Während des Trainings wird das Modell auf der Grundlage der Val-Datensätzen evaluiert. Wenn der Loss zunimmt oder die Genauigkeit abnimmt, wird das Training abgebrochen. [16]

Optimierer

Optimierer sind mathematische Funktionen, die von den erlernbaren Parametern des Modells, nämlich von den Gewichtungen und Biases, abhängen. Optimierer helfen dabei, die Gewichtungen und die Lernrate des neuronalen Netzes zu ändern, um die Summe der Fehler im Modell (Loss) zu minimieren. Der Gradient-Descent ist ein Optimierungsalgorithmus, der auf einer Konvexen Funktion basiert. Der optimiert die Parameter iterativ, um diese gegebene Funktion auf ihr lokales Minimum zu erreichen. Zudem minimiert der den Loss, indem er die Werte in die entgegengesetzte Richtung des steilsten Anstiegs ändert. Wie groß die Schritte der Änderungen in Richtung des lokalen Minimums sind, bestimmt dabei die Lernrate. Somit kann die Lernrate entscheiden, wie schnell die Funktion das lokalen Minimum erreicht. Diese Lernrate kann nicht zu groß oder zu klein sein, da sonst das lokalen Minimum verpasst wird oder das Optimieren zu lange dauert. [42]

2.1.4 Evaluationsmetriken

Eine wichtige Aufgabe bei der Erstellung eines Modells ist die Bewertung seiner Leistung. Zur Messung der Klassifizierungsleistung werden in der Arbeit der macro-averaged F1-Score und Accuracy (Genauigkeit) verwendet.

Der F1-Score wird für jede Klasse bei der multiclass Klassifizierung separat berechnet. Dieser ist eine Kombination aus Precision und Recall. TP_i *(true positives)* sind die Beispiele, die zur Klasse i gehören und richtig klassifiziert werden. TN_i *(true negatives)* sind die Beispiele, die zu den anderen Klassen gehören und richtig klassifiziert werden. FP_i *(false positives)* sind die Beispiele, die zu der Klasse i gehören, aber falsch klassifiziert werden. FN_i *(false negatives)* sind die Beispiele, die zur anderen Klassen gehören, aber falsch klassifiziert sind. [2, 4]

Der Macro-averaged F1-Score ist der durchschnittliche F1-Score aller Klassen, wobei der F1-Score jeder Klasse das gleiche Gewicht hat. Zudem ist N die Anzahl der Klassen. [4]

$$Precision = \frac{TP_i}{TP_i + FP_i} \tag{2.12}$$

$$Recall = \frac{TP_i + FN_i}{TP_i} \tag{2.13}$$

$$F1\text{-}Score_i = 2 * \frac{Precision_i * Recall_i}{Precision_i + Recall_i} \tag{2.14}$$

$$Macro\text{-}averaged\ F1\text{-}Score = \frac{\sum_{i=1}^{N} F1\text{-}Score_i}{N} \tag{2.15}$$

Die Genauigkeit ist die Wahrscheinlichkeit, dass die Modellvorhersage richtig ist. Sie misst den Prozentsatz der korrekt klassifizierten Beispiele an der Gesamtzahl der Beispiele. Die Formel der Genauigkeit berücksichtigt die Anzahl der *true positive* und *true negative* Beispiele im Zähler und die Anzahl aller Beispiele der Confusion-Matrix im Nenner. [27]

$$Accuracy\ = \frac{Correct\ predictions}{All\ predictions} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.16}$$

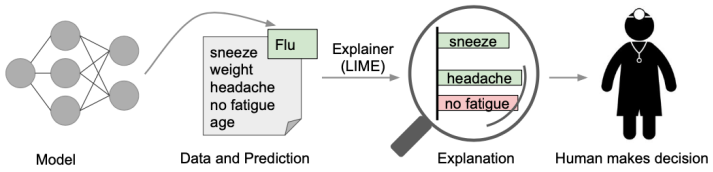
2.2 Erklärbare künstliche Intelligenz

Es gibt mehr und mehr beeindruckende Erfolge in den Bereichen der Verarbeitung natürlicher Sprache und KI-Anwendungen. Bei KI mit neuronalen Netzwerken wird oft die stochastische Optimierung eingesetzt. Dies macht die Entscheidungen dieser KI-Modelle für den Menschen schwer nachvollziehbar. KI-basierte Systeme haben beispielsweise deshalb einen großen Nachteil, nämlich eine mangelnde Transparenz. Viele Modelle haben eine gute Performanz und eine hohe Genauigkeit, können jedoch ihre Ausgaben nicht

erklären. Bei erklärbarer KI werden Modelle entwickelt, bei den die Menschen die Entscheidungen der Modelle verstehen können. Dann ist es zudem einfacher zu entscheiden, welches Modell gewählt werden sollte. Zudem spielt sie eine wichtige Rolle in vielen Bereichen wie Medizin und Finanzen. [38]

2.2.1 LIME

LIME ist eine Technik, die Prognosen eines beliebigen Klassifikators auf eine interpretierbare Weise erklären kann. LIME approximiert einen Klassifikator lokal mit einem interpretierbaren Modell. Für eine interpretierbare Erklärung muss die Repräsentation für den Menschen verständlich sein. Die folgende Abbildung 2.4 stellt den Prozess der Erklärung dar. Ein Arzt kann entscheiden, ob er dieser Vorhersage aufgrund der Erklärung der Vorhersage des Modells vertraut. In diesem Fall ist die Erklärung eine Liste der Symptome mit Gewichtung. Die Symptome „sneeze“ und „headache“ tragen zur Vorhersage „Flu“ bei und „no fatigue“ wirkt dagegen. [49]



Quelle: [49]

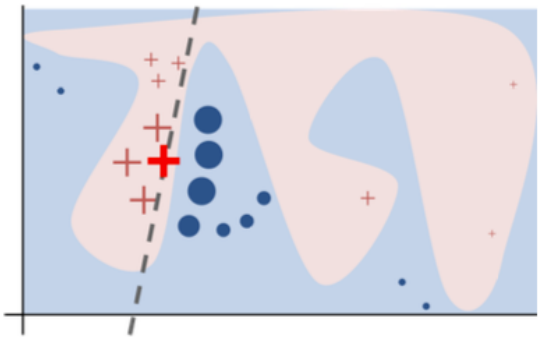
Abbildung 2.4: Erklärung einzelner Vorhersagen mit LIME

Bei LIME wird eine Erklärung als ein Modell $g \in G$ definiert, wobei G eine Klasse von potenziell interpretierbaren Modellen wie beispielsweise lineare Modelle ist. $\Omega(g)$ ist ein Maß für die Komplexität von g. $\Omega(g)$ ist beispielsweise für lineare Modelle die Anzahl der Gewichte ungleich Null. Bei Klassifikatoren ist $f(x)$ die Probabilität, dass x zu einer bestimmten Klasse gehört. $\pi_x(z)$ ist ein Maß für die Nähe von einer Instanz z zu x. Zudem definiert $\pi_x(z)$ die Lokalität von x. Eine Treue-Funktion $L(f, g, \pi_x)$ ist ein Maß dafür, wie untreu g bei der Approximation von f mit der durch π_x definierten Lokalität ist. In LIME wird $L(f, g, \pi_x)$ minimiert, um die Interpretierbarkeit und die lokale Treue zu gewährleisten. Während die Formel minimiert wird, muss $\Omega(g)$ niedrig genug sein, sodass g noch gut verständlich ist. Mithilfe der folgenden Formel 2.17 wird die Erklärung

generiert. [49]

$$\xi(x) = \underset{g \in G}{argmin} \quad L(f, g, \pi_x) + \Omega(g) \tag{2.17}$$

Die Folgende Abbildung 2.5 zeigt die Intuition von LIME. Die Entscheidungsfunktion f des Black-Box-Modells wird durch den blauen und rosa Hintergrund dargestellt. Diese Funktion ist komplex und durch ein lineares Modell nicht gut anzunähern. Das fette rote Kreuz ist das erläuterte Beispiel. Stichprobeninstanzen werden ausgewählt. Dann werden ihre Vorhersagen mit f erzeugt. Diese Instanzen werden anhand der Distanz zu der zu erklärenden Instanz gewichtet. Die gestrichelte Linie ist die erlernte lokale Erklärung. [49]



Quelle: [49]

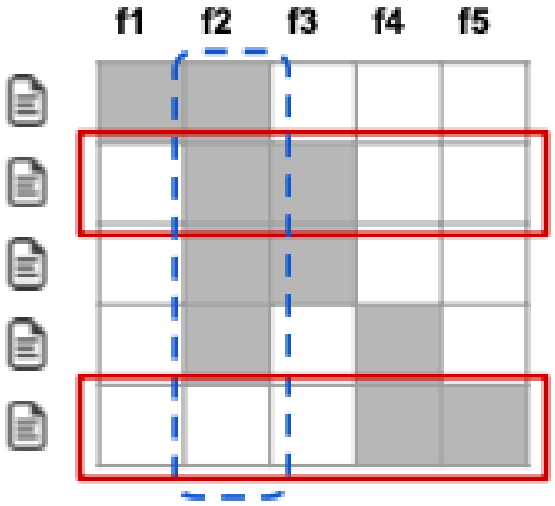
Abbildung 2.5: Intuition von LIME

2.2.2 SP-LIME

LIME bietet eine Erklärung für eine Vorhersage. Es reicht jedoch nicht aus, um ein ganzes Modell zu bewerten. In Submodular Pick Local Interpretable Model-agnostic Explanations (SP-LIME) wird eine Menge von einzelnen Instanzen interpretiert, wodurch SP-LIME ein globales Verständnis eines Modells liefert. Diese Instanzen müssen mit Bedacht gewählt werden, da es kostspielig ist, viele Erklärungen zu überprüfen. Das Budget B ist die Anzahl der Erklärungen, für deren Überprüfung man Zeit hat. X ist eine Menge der Instanzen. „Pick Step“ ist der Prozess der Auswahl von B Instanzen, die später geprüft werden sollen. [49]

Angesichts der Erklärungen für eine Menge von Instanzen $X(|X| = n)$ wird eine $n \times d'$ Erklärungsmatrix W erzeugt, die das lokale Gewicht der interpretierbaren Komponenten

für jede Instanz darstellt. I_j ist das globale Gewicht einer Spalte j von W. Die folgende Abbildung zeigt eine Erklärungsmatrix W mit $n = d' = 5$ für Texte, wobei W binär ist. Die Zeilen repräsentieren Instanzen(Dokumente) und die Spalten repräsentieren Merkmale(Wörter). Das Merkmal $f2$ hat die höchste Bedeutung, da es am häufigsten für Erklärungen der Instanzen verwendet wird. Auf der anderen Seite haben die Merkmale $f1$ und $f5$ den geringsten Einfluss. Die Instanzen 2 und 5 werden ausgewählt, da sie alle anderen Merkmale außer Merkmal $f1$ enthalten. [49]



Quelle: [49]

Abbildung 2.6: Intuition von SP-LIME

Bei der Auswahl der Instanzen wird vermieden, dass ähnliche Instanzen ausgewählt werden, sodass mehrere Merkmale einbezogen werden. In der Abbildung 2.6 haben z.B die zweite und dritte Instanz die gleichen Merkmale, sodass nur eine von ihnen ausgewählt wird. [49]

Die Intuition der nicht-redundanten Abdeckung wird in der Formel 2.18 dargestellt, wobei die Abdeckung als Mengenfunktion c definiert wird. Die Funktion c berechnet das Gesamtgewicht der Merkmale (Feature Importance), die in mindestens einer Instanz in einer Menge V vorkommen. [49]

$$c(V, W, I) = \quad \Sigma_{j=1}^{d'} \quad 1_{[\exists i \in V : W_{ij} > 0]} I_j \tag{2.18}$$

Das Pick-Problem ist durch die Formel 2.19 definiert. Mit dieser Formel wird die Menge V gefunden, mit welcher die höchste Abdeckung erreicht wird. [49]

$$Pick(W, I) = \underset{V,|V|\leq B}{argmax} c(V, W, I) \tag{2.19}$$

In der Formel 2.19 wird versucht, den Wert einer gewichteten Überdeckungsfunktion zu maximieren. $c(V \cup i, W, I) - c(V, W, I)$ ist der marginale Überdeckungsgewinn beim Hinzufügen einer Instanz i zu einer Menge V. Mit einem gierigen Algorithmus werden Instanzen iterativ zu einer Menge V mit dem höchsten marginalen Überdeckungsgewinn hinzufügt. [49]

3 Textvorverarbeitung

Textvorverarbeitung sind Techniken, deren Ziel es ist, Texte für Aufgaben der Verarbeitung natürlicher Sprache vorzubereiten. Textdaten im natürlichsprachlichen Format sind unstrukturiert und verrauscht. Beispielsweise gibt es Emojis, Satzzeichen und Rechtschreibfehler in Texten. Darüber hinaus können Modelle für maschinelles Lernen in den meisten Fällen nicht direkt mit Text arbeiten. Texte müssen vor dem Training in reine Zahlenwerten umgewandelt werden. Die Textvorverarbeitung bereinigt die Textdaten und wandelt Texte in Zahlenwerte um. Erfolgreiches maschinelles Lernen hängt oft von der richtigen Vorverarbeitung ab. [22, 50]

3.1 Tokenisierung

Die Tokenisierung ist häufig der erste Schritt der Textvorbereitung bei der Verarbeitung natürlicher Sprache. Das Hauptziel der Tokenisierung ist die Umwandlung der Texte in einen Strom von Verarbeitungseinheiten, sogenannte Token. Mit den erhaltenen Token wird ein Vokabular aufgebaut. Ein Vokabular ist eine Menge einzigartiger Wörter in einem Korpus. Außerdem enthält ein Vokabular nicht alle vorkommenden Wörter, sondern nur die K häufigsten Wörtern, wobei der K-Wert festgelegt werden kann. [29, 44]

Die Tokenisierung kann auf drei Ebenen durchgeführt werden: Wort-, Charakter- oder Subcharakterebene. Die am häufigsten verwendete Ebene ist die Wortebene. In diesem Fall werden die Texte basierend auf dem Trennzeichen in den Wörtern getrennt. Bei der Tokenisierung auf Wortebene ist es schwierig, mit Wörtern umzugehen, die nicht in dem Vokabular enthalten sind. Jedoch gibt es einen Trick dafür: die seltensten Wörter des Vokabulars werden durch unbekannte Token ersetzt. Auf diese Weise können auch diese Wörter bearbeitet werden. Allerdings verlieren die abgebildete Wörter dann alle Informationen. [44]

3.2 Normalisierung

Ein und dasselbe Wort hat unterschiedliche Formate. Zum Beispiel hat das Wort „love“ die folgenden zusätzlichen Formate: „love“, „loves“, „loved“. Diese Wörter haben in den meisten Fällen eine sehr ähnliche oder die gleiche Bedeutung. Die Normalisierung reduziert die Zufälligkeit eines Wortformats und bringt es in ein vordefiniertes Standardformat. Die Normalisierung umfasst die Erweiterung der Kontraktionen, die Entfernung von Satzzeichen und unerwünschten Zeichen, das Stemming und die Lemmatisierung. [68]

3.2.1 Erweiterung der Kontraktionen

Kontraktionen sind Wortkombinationen, die verkürzt werden, indem Buchstaben entfernt und durch ein Apostroph ersetzt werden. Beispielsweise kann die Kombination „do not“ zu „don’t“ verkürzt werden. Der Prozess der Umwandlung von „don’t“ in „do not“ wird als Erweiterung der Kontraktion bezeichnet. Dadurch werden Texte standardisiert. [68]

3.2.2 Stemming

Alle Wörter werden beim Stemming in ihren Wortstamm umgewandelt. Eine Nachschlagetabelle wird verwendet, um das Wort und seinen entsprechenden Wortstamm zu finden. Das Wort „love“ ist zum Beispiel ein Wortstamm von „loves“, „loved“, „loving“. Beim Stemming werden die letzten Buchstaben eines bestimmten Wortes entfernt, um eine kürzere Form zu erhalten, die möglicherweise keine Bedeutung hat. [52]

3.2.3 Lemmatisierung

Die Lemmatisierung ist eine fortgeschrittenere Form der Wortstammbildung. Bei der Lemmatisierung wird ein Wort auf sein Grundformat reduziert, das als „lemma“ bezeichnet wird. Das Grundformat des Wortes „laughing“ ist beispielsweise „laugh“. Um ein Grundformat eines Worts zu erkennen, müssen der Kontext und die Bedeutung des Worts analysiert werden. [25]

Im Vergleich zur Lemmatisierung wird beim Stemming keine Analyse über den Kontext und die Bedeutung des Worts durchgeführt. Daher können unterschiedliche Bedeutungen eines Worts in unterschiedlichen Kontexten beim Stemming nicht unterschieden werden.

Wegen der komplexeren Analyse kostet die Lemmatisierung mehr Zeit und erfordert mehr Rechenleistung. [50]

3.3 Textrepräsentation

3.3.1 One-Hot-Kodierung

Kategoriale Daten sind Variablen, die begrenzte festgelegte Label-Werte enthalten. Diese Werte stehen jeweils für eine andere Kategorie. Kategoriale Werte müssen in den meisten Verfahren des maschinellen Lernens auf numerische Werte abgebildet werden. Manchmal reicht es jedoch nicht aus, dass jeder einzigartiger Label-Wert jeweils einem ganzzahligen Wert zugeordnet wird, wenn diese Werte in keiner geordneten Beziehung zueinander stehen. Wenn beispielsweise die Farben „Rot“, „Grün“ und „Blau“ jeweils auf 1, 2, 3 kodiert werden, wird impliziert angenommen, dass diese Farben eine Reihenfolge haben. Um diese implizierte Ordnung zu vermeiden, werden bei der One-Hot-Kodierung diese ganzzahlig kodierte Werte auf neue Binärwerte abgebildet. Nach der One-Hot-Kodierung werden diese Farben jeweils auf 001, 010 und 001 abgebildet. Die folgende Abbildung 3.1 zeigt ein Beispiel für die One-Hot-Kodierung. [18]

	Rot	Grün	Blau
Rot	1	0	0
Grün	0	1	0
Blau	0	0	1

Abbildung 3.1: Ein Beispiel der One-Hot-Kodierung

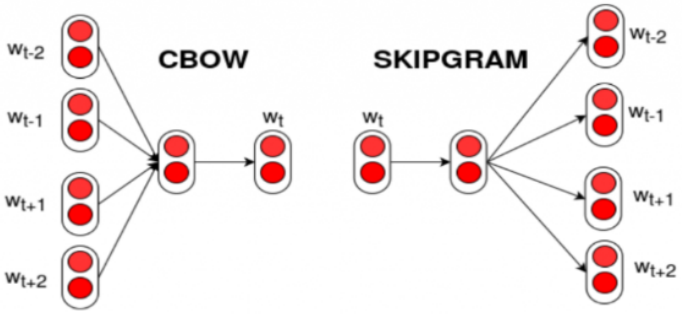
3.3.2 Worteinbettung

Bei der Worteinbettung (Word Embedding) können die Wörter jeweils einem Vektor zugeordnet werden. Diese Methode ermöglicht es, dass Wörter mit ähnlichen Bedeutungen

eine ähnliche Darstellung erhalten. Eine Worteinbettungsschicht kann mit ihrem eigenen Netz trainiert werden. Zudem gibt es vortrainierte Worteinbettungsschichten, deren Gewichte von anderen Modellen übernommen werden können. Word2Vec und GloVe sind zwei Arten, mit den Worteinbettungen trainiert werden. [16]

Word2Vec

Word2Vec ist ein zweischichtiges neuronales Feedforward-Netzwerkmodell. Es gibt zwei Modellarchitekturen: das Continuous Bag of Words (CBOW)-Modell und das Skip-Gram-Modell. Das Skip-Gram-Modell nimmt jedes Wort im Korpus als Eingabe und sagt die Kontextwörter voraus, während das CBOW-Modell die Kontextwörter als Eingabe verwendet und das ursprüngliche Wort vorhersagt. Die Beziehungen zwischen den Wörtern ergeben sich aus den Kosinus-Distanzen zwischen den Wörtern. Der Abstand zwischen „Mädchen“ und „Frauen“ ist ungefähr gleich wie der Abstand zwischen „Jungen“ und „Männer“. Dies ermöglicht es, dass das Ergebnis von „Jungen + Frauen - Mädchen “ ungefähr gleich „Männer“ ist. [1, 40]



Quelle: [1]

Abbildung 3.2: CBOW-Modell vs Skip-Gram-Modell

GloVe

GloVe steht für Global Vertors, weil das GloVe-Modell globale Korpusstatistiken berücksichtigt. Dieses Modell basiert auf Matrixfaktorisierungsverfahren für die Wort-Kontext-Matrix. Es wird eine Matrix des gemeinsamen Auftretens von Wörtern erstellt, bei der jede Zeile einem Wort und jede Spalte einem Kontextwort entspricht [33]. Die Häufigkeit des gleichzeitigen Auftreten jedes Wortes mit jedem Kontextwort wird berechnet. Da der globale Kontext häufig groß ist, wird diese Matrix gemäß der in der folgenden Abbildung 3.3 dargestellten Formel faktorisiert. Daraus ergibt sich eine kleinere Worte-Merkmale-Matrix, in der jede Zeile ein Wort im Form eines Vektors darstellt. GloVe ist zudem eine

Erweiterung von Word2Vec und ermöglicht auch die Berechnung des Abstands zwischen Wörtern. [1, 16]

Abbildung 3.3: GloVe-Matrixformel der Faktorisierung

4 Datengrundlage

Der in der Arbeit verwendete Datensatz ist ein Datensatz von Googel zur Klassifizierung von Emotionen, nämlich GoEmotions [48]. Dieser Datensatz enthält 58000 Reddit-Kommentare, die aus beliebten englischsprachigen Subreddits extrahiert und mit einer oder mehreren Klassen von 27 Emotionsklassen und einer neutralen Klasse beschriftet wurden. Diese 27 Emotionsklassen können in 6 Ekman-Emotionsklassen eingeordnet werden, nämlich Anger, Disgust, Fear, Joy, Sadness und Surprise. In dieser Arbeit wird dieser Datensatz in Ekman-Emotionsklassen klassifiziert. Zudem werden nur die Einträge mit einem Label betrachtet. Der Datensatz wird in Training-Datensatz, Val-Datensatz und Test-Datensatz aufgeteilt, jeweils 80%, 10% , 10%.

Die Verteilung der Klassen des Training-Datensatzes wird durch die folgende Abbildung dargestellt:

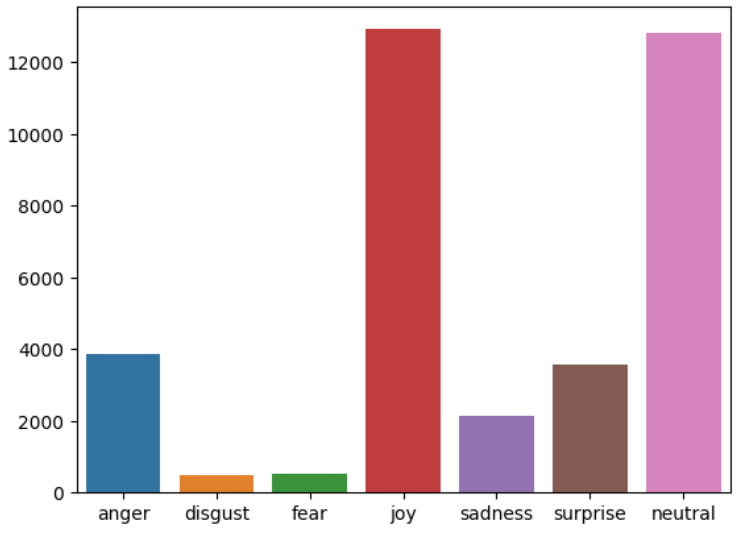
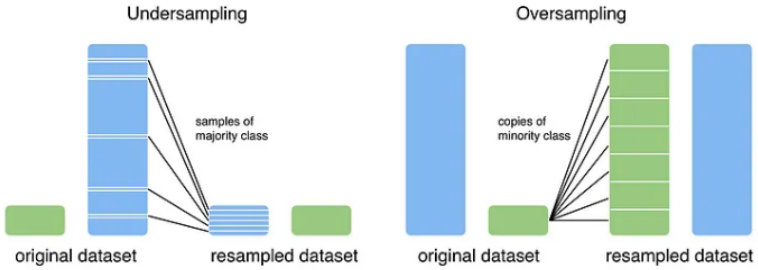


Abbildung 4.1: Aufteilung der Klassen des Training-Datensatzes

4.1 Datensatz balancieren

Der Training-Datensatz ist nicht balanciert, weil die Einträge mit dem Label Joy oder Neutral mehr als die Hälfte der Gesamteinträge ausmachen. Infolgedessen sind die Klassen Joy und Neutral überrepräsentiert. Gleichzeitig sind die anderen Klassen unterrepräsentiert, insbesondere die Klassen Disgust und Fear.

Das Hauptproblem bei der Klassifizierung der nicht balancierten Datensätzen besteht darin, dass die Mehrheitsklasse einen größeren Einfluss auf ein Modell hat. Allerdings ist die Minderheitsklasse genauso wichtig wie die Mehrheitsklasse. Methoden zum Balancieren von Datensätzen sind das Over-Sampling der Minderheitsklasse und das Under-Sampling der Mehrheitsklasse. Beim Over-Sampling werden Elemente einer Minderheitsklasse nach dem Zufallsprinzip kopiert. Andererseits werden beim Under-Sampling Elemente einer Klasse nach dem Zufallsprinzip gelöscht. Die folgende Abbildung 4.2 stellt den Prozess von Over-Sampling und Under-Sampling dar. [39]



Quelle: [37]

Abbildung 4.2: Over-Sampling vs Under-Sampling

Für diese Arbeit bedeutet dies, dass die Elemente der Klassen Joy und Neutral nach dem Zufallsprinzip gelöscht werden. Hingegen werden die Elemente aus der Klasse Anger, Disgust, Fear, Sadness und Surprise nach dem Zufallsprinzip kopiert. Schlussendlich ist die Anzahl jeder Klasse 4000. Der Training-Datensatz enthält daher 28000 Texte. Die Tabelle 4.1 zeigt die Emotionsklassen-Verteilung in Training-Datensatz vor und nach der Balancierung.

Training-Datensatz	Anger	Disgust	Fear	Joy	Sadness	Surprise	Neutral	Total
Vor Balancierung	3877	497	515	12913	2121	3528	12717	36168
Nach Balancierung	4000	4000	4000	4000	4000	4000	4000	28000

Tabelle 4.1: Training-Datensatz vor und nach der Balancierung

Der Val-Datensatz und der Test-Datensatz sind nicht balanciert und enthalten jeweils etwa 4500 Texte. Die Verteilung der Klassen des Val-Datensatzes und des Test-Datensatzes sind ähnlich wie die Abbildung 4.1 darstellt. Die folgende Tabelle 4.2 zeigt jeweils die Anzahl der Texte pro Emotionsklasse des Training-, Val- und Test-Datensatzes.

Datensatz	Anger	Disgust	Fear	Joy	Sadness	Surprise	Neutral	Total
Training	4000	4000	4000	4000	4000	4000	4000	28000
Val	485	61	66	1666	241	428	1580	4527
Test	520	76	77	1601	259	447	1596	4576

Tabelle 4.2: Emotionsklassen-Verteilung in Datensätzen

4.2 Textvorbereitung

Die Elementen des Datensatzes werden bereinigt. Die Emojis, Links und HTML-Tags, Interpunktionen, Sonderzeichen, Zahlen und zusätzliche Leerzeichen werden gelöscht. Zwischen zwei Wörtern steht immer nur ein Leerzeichen. Alle Wörter werden in Kleinbuchstaben geschrieben. Darüber hinaus werden alle Kontraktionen erweitert. Zudem werden alle Wörter lemmatisiert, sodass sie auf ihr Grundformat reduziert werden.

Die Texten werden auf Wortebene tokenisiert. Denn durch die Tokenisierung auf Wortebene lässt sich leicht erklären, ob ein Modell Vorhersagen auf der Grundlage aussagekräftigen Daten macht. Das Prepadding wird bei jedem tokenisierten Text durchgeführt, damit jeder tokenisierte Text die gleiche Länge besitzt. Der längste Text enthält 40 Wörter. Die meisten Texte bestehen aus 5 bis 35 Wörtern. Durch Prepadding werden alle Arrays von tokenisierten Texten, die kürzer als 40 Wörter sind, vorne mit 0 aufgefüllt, sodass alle Arrays die Länge 40 haben. Hier wird Prepadding anstelle von Postpadding verwendet, sodass jeder tokenisierte Text am Ende jedes Eingabearrays steht und so es für ein LSTM-Netz leicht zu merken ist. Außerdem werden alle Emotionsklassen mit One-Hot-Kodierung kodiert.

Jedes Wort aus jedem tokenisierten Text wird dann einem Vektor aus der Worteinbettung zugeordnet. In der Arbeit werden vortrainierte Worteinbettungen untersucht. Der verwendete Training-Datensatz umfasst insgesamt 12864 Wörter. Je nachdem welcher vor-trainierte Datensatz verwendet wird, gibt es verschiedene GloVe-Worteinbettungen [58]. Eine davon wurde mit den Texten aus Wikipedia 2014 und Gigaword 5 trainiert, die 6 Milliarden Token enthält. Die Größe ihrer Vektoren ist 300. Diese GloVe-Worteinbettung

wird in der Arbeit als GloVe-6B bezeichnet. Eine weitere GloVe-Worteinbettung wurde mit den Texten von Twitter trainiert, die 27 Milliarden Token enthalten. Die Größe Jedes Vektors dieser Worteinbettung ist 200. Diese Worteinbettung wird im Folgenden GloVe-Tweets genannt. Die Word2Vec-Worteinbettung [5] wurde von Google auf der Grundlage des Google-News-Datensatzes trainiert. Die Größe dieser Vektoren ist 300. GloVe-6B enthält 11623 Wörter des in der Arbeit verwendeten Datensatzes, während GloVe-Tweets 11915 Wörter des Datensatzes enthält. Zudem beinhaltet Word2Vec 11409 Wörter des Datensatzes. Daher enthählt GloVe-Tweets die meisten Wörter des verwendeten Datensatzes.

Worteinbettung	Vektorgröße	Anzahl der Wörter aus dem verwendeten Datensatz
GloVe-6B	300	11623
GloVe-Tweets	200	11915
Word2Vec	300	11409

Tabelle 4.3: Dimensionen und Anzahl der Wörter aus in der Arbeit verwendeten Datensatz der Worteinbettungen

5 Experimente und Resultate

In diesem Kapitel werden verschiedene Modelle mit unterschiedlicher Struktur trainiert. Es gibt 5 Arten von Modellen: LSTM, CNN, LSTM-CNN, CNN-LSTM und Ensemble. Jeder Typ hat wiederum mehrere Varianten. Jeder Typ wird unter seinen eigenen Varianten verglichen.

5.1 Aufbau der Experimente

In diesem Abschnitt wird beschrieben, unter welchen allgemeinen Bedingungen die folgenden Experimente durchgeführt werden.

Worteinbettung
In der Arbeit werden drei verschiedene Worteinbettungen untersucht, nämlich GloVe-6B, GloVe-Tweets und Word2Vec. Wenn ein Wort in einer Worteinbettung vorhanden ist, wird das Wort durch den Vektor des Wortes der Worteinbettung repräsentiert. Sonst wird das Wort durch einen Vektor repräsentiert, der mit 0 aufgefüllt ist. Diese Wortvektoren werden während des Trainings nicht verändert. Da die maximale Textlänge des Datensatzes 40 beträgt und die Größe der Worteinbettungsvektoren entweder 200 oder 300 ist, wird jeder Text bei GloVe-Tweets einem $40 * 200$ Vektor und bei GloVe-6B oder Word2Vec einem $40 * 300$ Vektor zugewiesen.

Batch-Size und Epochen
Die Batch-Size ist die Anzahl der verarbeiteten Beispiele, bevor das Modell aktualisiert wird. Normalerweise wird die Batch-Size als eine Zweierpotenz im Bereich zwischen 16 und 512 gewählt [70]. Die Anzahl der Epochen ist die Anzahl der vollständigen Durchläufe durch den Trainingsdatensatz [19]. Für alle Experimente beträgt die Batch-Size 32. Außerdem ist die Anzahl der Epochen 16.

Early Stopping
Wenn sich das Loss des Val-Datensatzes (val_loss) eines Modells nach 3 Epochen nicht verbessert hat, wird das Training früher beendet.

Dropout
Die Dropout-Rate der Experimente beträgt 0,5, da die Untersuchung von Srivastava et al. zeigt, dass die optimale Dropout-Rate bei etwa 0,5 liegt [57].

Optimierer
Diese Experimente verwenden Adaptive Moment Estimation (Adam) als Optimierer. Mit Adam werden die Gewichtungen eines Netzes iterative basierend auf dem Training-Datensatz angepasst. Aus der Schätzung der ersten und zweiten Momente der Gradienten berechnet Adam individuelle adaptive Lernraten für Parameter. Die Standardeinstellungen sind $lerning\ rate = 0,001, \beta_1 = 0,9, \beta_2 = 0,999$ und $\epsilon = 10^{-8}$, wobei β_1 und β_2 jeweils die exponentielle Decay-Rate für die Schätzung des ersten und zweiten Moments sind. ϵ ist eine sehr kleine Zahl, um eine Division durch Null zu vermeiden. Für die folgenden Experimente werden die Standardeinstellungen verwendet. [15, 36]

Checkpoint
Beim Training eines Modells wird nach jeder Epoche geprüft, ob dieses Modell einen besseren val_loss erreicht hat. Wenn ja, dann wird dieses Modell gespeichert. Am Ende wird das Modell mit dem kleinsten val_loss ausgewählt, das als das beste Modell einer Variante eines Typs verwendet wird.

Evaluation
Anhand des aus dem Test-Datensatz erhaltenen macro-averaged F1-Scores werden die Modelle bewertet und miteinander verglichen. Der F1-Score wird verwendet, weil der Test-Datensatz nicht balanciert ist. Darüber hinaus berücksichtigt der F1-Score, wie die Daten verteilt sind. Jedoch ist der F1-Score schwer zu interpretieren. Die Genauigkeit ist verständlicher als der F1-Score. Daher werden die Modelle auch auf der Grundlage der erzielten Genauigkeit verglichen. Die Genauigkeit ist jedoch nicht für nicht balancierte Datensätze geeignet [59]. Das Kriterium eines besseren Modells ist daher der macro-averaged F1-Score.

5.2 LSTM-Modell

5.2.1 Architektur

Es wird im folgenden beschrieben, wie die Architektur des LSTM-Modells aussieht. Diese wird in der Abbildung 5.1 dargestellt.

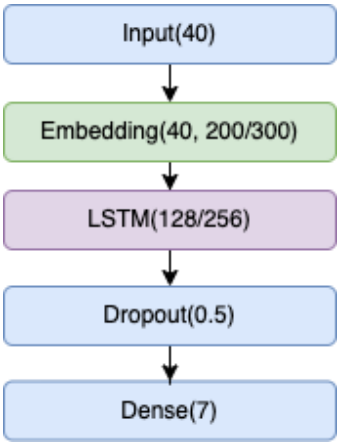


Abbildung 5.1: LSTM-Modell Architektur

Input-Schicht
Jede Eingabe ist ein vorab aufgefülltes (prepadded) Array mit einer Länge von 40, weil der längste Text 40 Wörter enthält.

Embedding-Schicht
Hier werden 3 Typen der Worteinbettung untersucht, nämlich GloVe-Tweets, GloVe-6B und Word2Vec. Da die Größe der Vektoren von GloVe-Tweets 200 ist und die Größe der Vektoren von GloVe-6B und Word2Vec jeweils 300 beträgt, gibt es 2 Größen für die Worteinbettung: (40, 200) oder (40,300).

LSTM-Schicht
In den Experimenten werden die Leistungen des LSTM-Modells mit 128 Units und 256 Units verglichen, wobei die Anzahl von Units die Größe der Ausgabe ist. Diese LSTM-Schicht verwendet den Variational-RNN-Dropout (siehe Dropout in Abschnitt 2.1.3). Daher werden bei jedem Zeitschritt dieselben Knoten aus dem Netzwerk herausgenommen. Diese Dropout-Rate ist auf 0,5 festgelegt. Diese LSTM-Schicht gibt die letzte Ausgabe der Ausgabereihenfolge zurück.

Dropout-Schicht
Die Dropout-Rate dieser Dropout-Schicht beträgt ebenfalls 0,5, um das Overfitting zu verhindern.

Dense-Schichte
Es gibt 7 Typen von Emotionsklassen. Daher enthält diese Dense-Schichte 7 Neuronen. Die Aktivierungsfunktion ist Softmax. Die Ausgabe dieser Schicht ist die Wahrscheinlichkeiten der Emotionsklassen.

5.2.2 Resultate

In der Tabelle 5.1 sind die Ergebnisse der Experimente der verschiedenen LSTM-Modellvarianten aufgeführt.

Embedding	Embedding size	LSTM units	Accuracy(%)	F1 score(%)
GloVe-Tweets	200	128	58,30	52,03
GloVe-6B	300		55,33	50,93
Word2Vec	300		60,23	54,86
GloVe-Tweets	200	256	58,35	52,39
GloVe-6B	300		59,51	53,59
Word2Vec	300		59,62	53,47

Tabelle 5.1: Resultate der Experimente der LSTM-Modellvarianten

Das LSTM-Modell mit 128 Units und der Word2Vec-Worteinbettung hat die beste Leistung mit einer Genauigkeit von 60,23 % und einem F1-Score von 54,86% . Bei einer Verdoppelung der LSTM-Units schnitt das Word2Vec-Modell nicht besser ab, während die Modelle mit der GloVe-Worteinbettung eine bessere Leistung erbrachten. Unter den Modellen mit der gleichen Anzahl von Units weist ein Word2Vec-Modell die höchste Genauigkeit auf.

Das LSTM-Modell mit 256 Units und der Worteinbettung GloVe-6B ist das zweitbeste LSTM-Modell mit einer Genauigkeit von 59,51 % und einem F1-Score von 53,59%. Der F1-Score dieses Modells ist um etwa 3% höher als der des Modells mit 128 Units und der gleichen Einbettung. Das Modell mit 128 Units und der GloVe-Tweets-Worteinbettung hat eine Genauigkeit von 58,30% und einen F1-Score von 52,03% erreicht, und hat damit eine bessere Leistung als das Modell mit 128 Units und der GloVe-6B-Worteinbettung.

Allerdings hat dieses Modell mit der GloVe-Tweets-Worteinbettung fast die gleiche Leistung wie das Modell mit 256 Units und derselben Einbettung.

In der Abbildung 5.2 ist zu erkennen, dass bei dem besten LSTM-Modell nach 4 Epochen Overfitting stattfindet. Das beste Modell wird nach 8 Epochen gespeichert, da es den höchsten F1-Score auf dem Val-Datensatz erreicht hat. Das Training wurde nach 11 Epochen aufgrund eines Early-Stoppings abgebrochen.

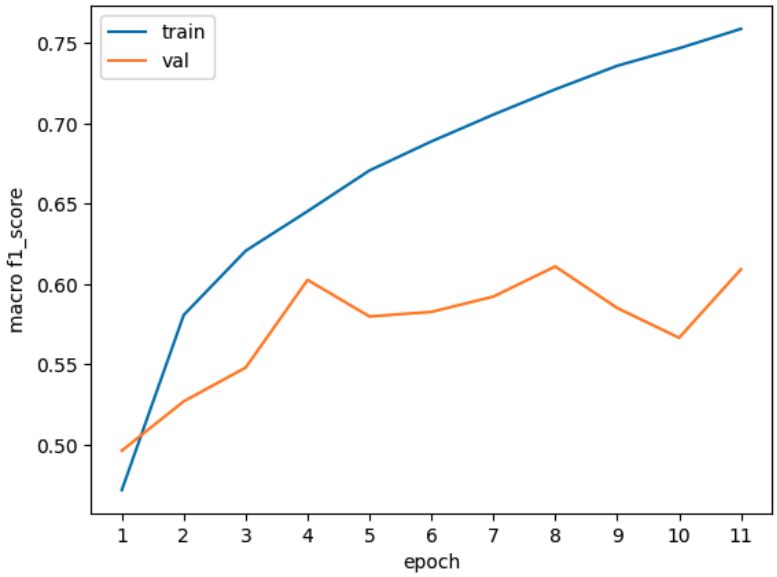


Abbildung 5.2: F1-Score des 128-Units-LSTM-Modells mit Word2Vec

5.3 CNN-Modell

5.3.1 Architektur

Die Arbeit „Convolutional Neural Networks for Sentence Classification“ [35] von Kim zeigt, dass ein CNN-Modell gut für die Textklassifikation geeignet ist. Alshubaily et al. [11] entwickelten ein CNN-Modell für die Textklassifikation mit einer Spatial-Dropout-Schicht auf der Einbettung des Eingabesatzes. Die Architektur des CNN-Modells orientiert sich an der Arbeit von Alshubaily et al. und der Arbeit von Kim. In der Abbildung 5.3 wird diese Architektur dargestellt. Der Aufbau von Eingabe-, Einbettung-, Dropout- und Dense-Schicht ist gleich wie beim LSTM-Modell.

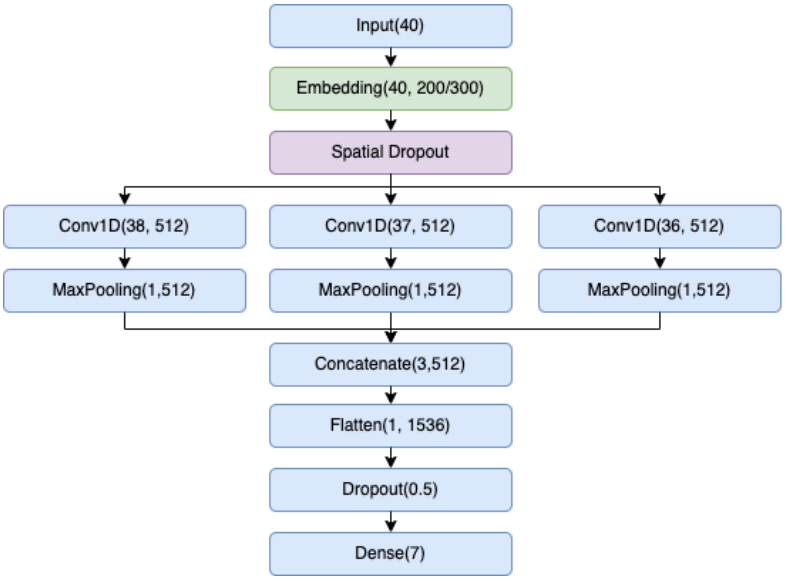


Abbildung 5.3: CNN-Modell Architektur

Spatial-Dropout-Schicht

Bei dieser Schicht werden ganze 1D-Feature-Maps anstelle einzelner Elemente gelöscht [60]. Die Dropout-Rate dieser Schicht ist 0,5. Die Rolle des Dropouts besteht darin, die Generalisierungsleistung zu verbessern, indem ein Übertraning verhindert wird [11].

Convolutional-Schicht

Jeder Text wird in einer 2D-Matrix dargestellt. Daher werden 1D-Convolutional-Schichten verwendet, bei denen sich alle Kernel jedes Mal einen Schritt entlang der Dimension der Wörter bewegen. Es gibt in dem Modell 3 parallele 1D-Convolutional-Schichten mit den Filtergrößen 3, 4 und 5. Jede Schicht hat 512 Filter und erzeugt damit 512 Feature-Maps. Dadurch können verschiedene Merkmale aus einem Text gelernt werden.

MaxPooling-Schicht

Auf jede 1D-Convolutional-Schicht folgt eine MaxPooling-Schicht. Beim MaxPooling wird das größte Element aus jeder Feature-Map ausgewählt. Somit besteht die Ausgabe nach der MaxPooling-Schicht aus den auffälligsten Merkmalen der Feature-Maps.

Concatenate-Schicht

Die Ausgaben der einzelnen MaxPooling-Schichten werden miteinander konkateniert. Das Ergebnis ist eine $3 * 512$ 2D-Matrix.

Flatten-Schicht
Die 2D-Matrix aus der Concatenate-Schicht wird in eine 1D-Matrix umgewandelt. Der Grund dafür ist, dass diese Matrix als Eingabe für die Dense-Schicht verwendet wird.

5.3.2 Resultate

Die Resultate der CNN-Modellvarianten werden in diesem Abschnitt beschrieben. Das Word2Vec-CNN-Modell mit einer Spatial-Dropout-Schicht hat die höchste Genauigkeit von 61,91% und den höchsten F1-Score von 55,97%. Das Word2Vec-CNN-Modell hat die beste Leistung unter den Modellen ohne einer Spatial-Dropout-Schicht sowie mit einer Spatial-Dropout-Schicht. Das GloVe-6B-CNN-Modell ist das leistungsschwächste Modell sowohl unter den Modellen mit einer Spatial-Dropout also auch unter den Modellen ohne einer Spatial-Dropout-Schicht. Eine Spatial-Dropout-Schicht hat den F1-Score aller Modelle um etwa 1% erhört.

Embedding	Embedding dimension	Spatial Dropout	Accuracy(%)	F1 score(%)
GloVe-Tweets	200	Yes	60,16	54,99
GloVe-6B	300		56,99	53,26
Word2Vec	300		61,91	55,97
GloVe-Tweets	200	No	59,64	53,50
GloVe-6B	300		56,73	52,19
Word2Vec	300		60,38	54,70

Tabelle 5.2: Resultate der Experimente der CNN-Modellvarianten

Die Abbildung 5.4 zeigt während des Trainings die Änderung des F1-Scores des Word2Vec-CNN-Modells mit einer Spatial-Dropout-Schicht. Das Modell erreicht den besten F1-Score des Val-Datensatzes in der 6. Epoche. Der F1-Score auf dem Training-Datensatz ist ab der 7. Epoche über 75% und hat sich immer weiter verbessert. Allerdings hat der F1-Score auf dem Val-Datensatz sich ab der 7. Epoche nicht weiter verbessert. Das Training wurde in der 8. Epoche aufgrund eines Early-Stopping abgebrochen.

Andererseits hat das LSTM-Modell in der 8. Epoche den besten F1-Score des Val-Datensatzes erzielt. Im Vergleich zum LSTM-Modell benötigt das CNN-Modell zwei Epochen weniger, um dessen besten F1-Score auf dem Val-Datensatz zu erreichen. Dies kann daran liegen, dass dieses LSTM-Modell die Hälfte Units weniger als dieses CNN-Modell hat. Darüber hinaus hat das CNN-Modell eine bessere Genauigkeit auf dem Test-Datensatz erreicht, die etwa 2% höher als die des LSTM-Modells ist. Der beste F1-Score

des CNN-Modells auf dem Test-Datensatz ist zudem um 1% höher als der des LSTM-Modells. Die beiden Modelle haben die beste Leistung mit der Word2Vec-Worteinbettung erreicht.

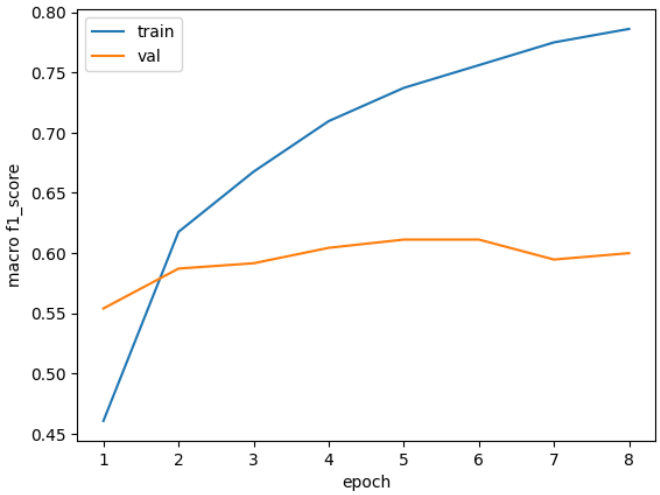


Abbildung 5.4: F1-Score des Word2Vec-CNN-Modells mit Spatial-Dropout

5.4 LSTM-CNN-Modell

5.4.1 Architektur

Joshi et al. [34] fanden in ihrer Arbeit aus, dass ein LSTM-CNN-Modell im Vergleich zu CNN-, LSTM- und CNN-LSTM-Modellen eine höhere Genauigkeit bei der Erkennung von Emotionen aufweist. Die folgende Architektur orientiert sich an dieser Arbeit von Joshi et al. Zudem kombiniert sie die Struktur des LSTM-Modells und des CNN-Modells der Arbeit. Die LSTM-Schicht liegt vor der CNN-Schicht. Ziel der Kombination dieser beiden Modelle besteht darin, dass die CNN-Schicht die Merkmale der wichtigen von der LSTM-Schicht erfassten Informationen lernt.

Dieses LSTM-CNN-Modell hat eine Schicht mehr als das CNN-Modell, nämlich eine LSTM-Schicht. Die anderen Schichten dieses LSTM-CNN-Modells sind genau dieselben wie die des CNN-Modells der Arbeit. Daher werden diese Schichten nicht noch einmal erläutert.

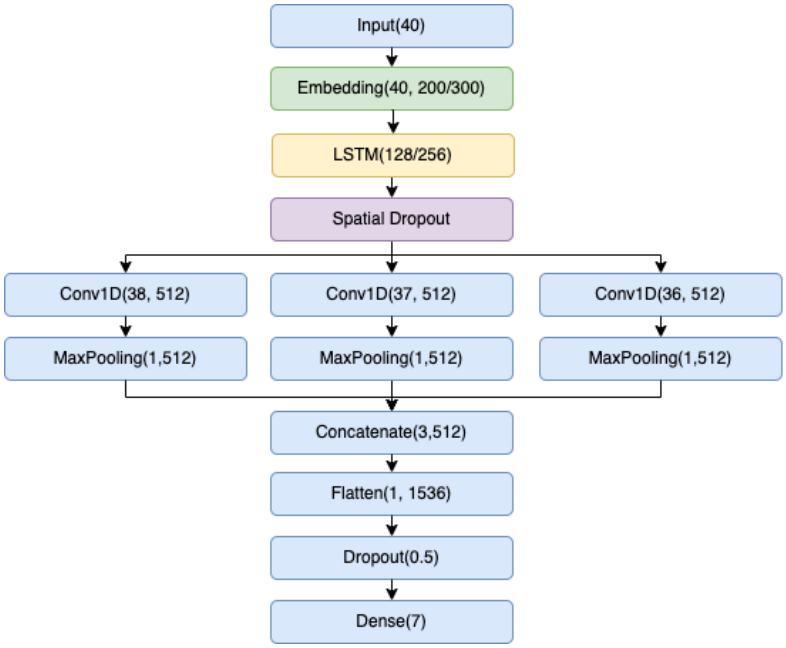


Abbildung 5.5: LSTM-CNN-Modell Architektur

LSTM-Schicht

Diese LSTM-Schicht liefert die vollständige Ausgabesequenz, während die LSTM-Schicht im LSTM-Modell nur die letzte Ausgabe zurückgibt. Der Grund dafür ist, dass die nachfolgende CNN-Schicht wichtige Merkmale aus dieser vollständigen Ausgabesequenz lernen kann. Diese LSTM-Schicht verwendet den Variational-RNN-Dropout mit der Dropout-Rate 0,5. Die Leistungen des LSTM-CNN-Modells mit 128 Units und 256 Units werden verglichen.

5.4.2 Resultate

In der Tabelle 5.3 sind die Resultate der Experimente der 12 LSTM-CNN-Modellvarianten aufgeführt. Das beste Modell ist das GloVe-Tweets-Modell mit 256 LSTM-Units ohne einer Spatial-Dropout-Schicht, das eine Genauigkeit von 62,61% und einen F1-Score von 57,14% aufweist.

Mit einer Spatial-Dropout-Schicht hat sich nur die Genauigkeit des GloVe-6B-Modells mit 256 LSTM-Units verbessert. Bei den anderen Modellen ist sowohl die Genauigkeit also auch der F1-Score gesunken.

Das Word2Vec-Modell hat die beste Leistung unter den Modellen mit 128 LSTM-Units, während das GloVe-Tweets-Modell das leistungsfähigste unter den Modellen mit 256 LSTM-Units ist.

Durch die Verdoppelung der LSTM-Units haben sich die Genauigkeiten der Modelle mit einer Spatial-Dropout-Schicht erhöht. Außerdem ist der F1-Score des GloVe-Tweets- und GloVe-6B-Modells mit einer Spatial-Dropout-Schicht gestiegen. Jedoch ist der F1-Score des Word2Vec-Modells mit einer Spatial-Dropout-Schicht gesunken. Bei den Modellen ohne einer Spatial-Dropout-Schicht verbesserte die Verdoppelung von LSTM-Units die Genauigkeit des GloVe-Tweets-Modells und den F1-Score des GloVe-Tweets- und GloVe-6B-Modells.

Embedding	Embedding size	LSTM Units	Spatial Dropout	Accuracy(%)	F1 score(%)
GloVe-Tweets	200	128	Yes	58,54	52,73
GloVe-6B	300			59,55	53,14
Word2Vec	300			60,47	55,27
GloVe-Tweets	200		No	59,83	54,40
GloVe-6B	300			59,94	54,77
Word2Vec	300			62,41	56,40
GloVe-Tweets	200	256	Yes	61,49	55,22
GloVe-6B	300			60,80	54,93
Word2Vec	300			60,53	54,20
GloVe-Tweets	200		No	62,61	57,14
GloVe-6B	300			59,75	54,95
Word2Vec	300			61,28	56,15

Tabelle 5.3: Resultate der Experimente der LSTM-CNN-Modellvarianten

Die Abbildung 5.6 stellt die Entwicklung des F1-Scores des GloVe-Tweets-Modells mit 256 LSTM-Units ohne einer Spatial-Dropout-Schicht dar, das das beste LSTM-CNN-Modell ist. Dieses Modell hat den höchsten F1-Score des Val-Datensatzes nach der 5. Epoche erreicht. Anschließend ist der F1-Score des Val-Datensatzes gesunken. Der F1-Score des Training-Datensatzes ist stets gestiegen und liegt nach der 8. Epoche bei 80%.

Das LSTM-CNN-Modell hat den höchsten F1-Score beim Val-Datensatz 3 Epochen früher erreicht als das LSTM-Modell und eine Epoche früher als das CNN-Modell erreicht. Der erreichte F1-Score beim Test-Datensatz ist 1% höher als der des CNN-Modells. Für das LSTM-Modell und das CNN-Modell sind die Modelle mit der Word2Vec-Worteinbettung die Leistungsfähigsten, während das beste LSTM-CNN-Modell mit der GloVe-Tweets-Worteinbettung ist.

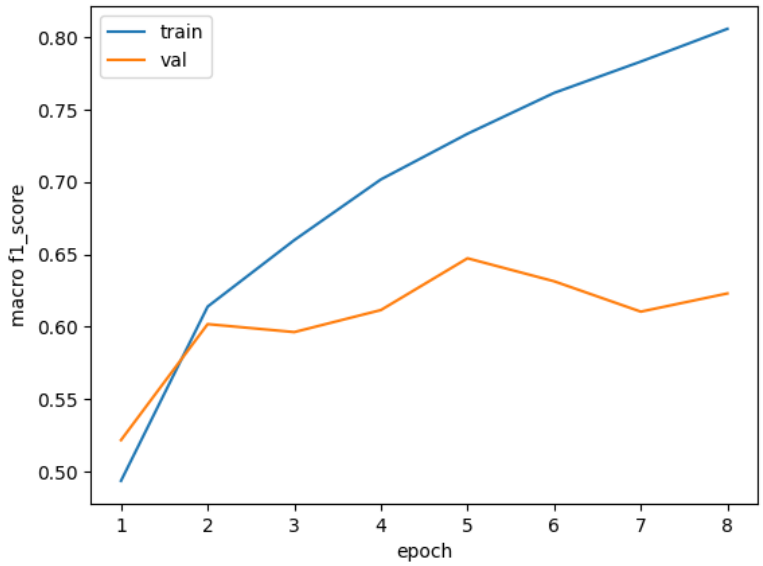


Abbildung 5.6: F1 Score des GloVe-Tweets-LSTM-CNN-Modells mit 256 LSTM-Units ohne Spatial-Dropout

Das beste LSTM-Modell ist das mit 128 LSTM-Units. Allerdings ist das best LSTM-CNN-Modell mit 256 LSTM-Units. Darüber hinaus ist das beste CNN-Modell mit einer Spatial-Dropout-Schicht, während das beste LSTM-CNN-Modell ohne einer Spatial-Dropout-Schicht ist.

5.5 CNN-LSTM-Modell

5.5.1 Architektur

Cahyani et al. [20] erstellten ein textbasiertes Emotionserkennungsmodell unter Verwendung eines CNN-BiLSTM-Modell. Mit einem CNN-Modell können Merkmale aus Texten extrahiert werden. Die Ausgabe eines LSTM-Modells behält die logische Reihenfolge der Wörter in einem Text bei und ermöglicht, irrelevante Informationen zu vergessen. Durch die Kombination dieser beiden Modelle können die Merkmale, die mit einer CNN-Schicht gelernt wurden, als Eingabe für eine LSTM-Schicht verwendet und von dieser Schicht erfasst werden.

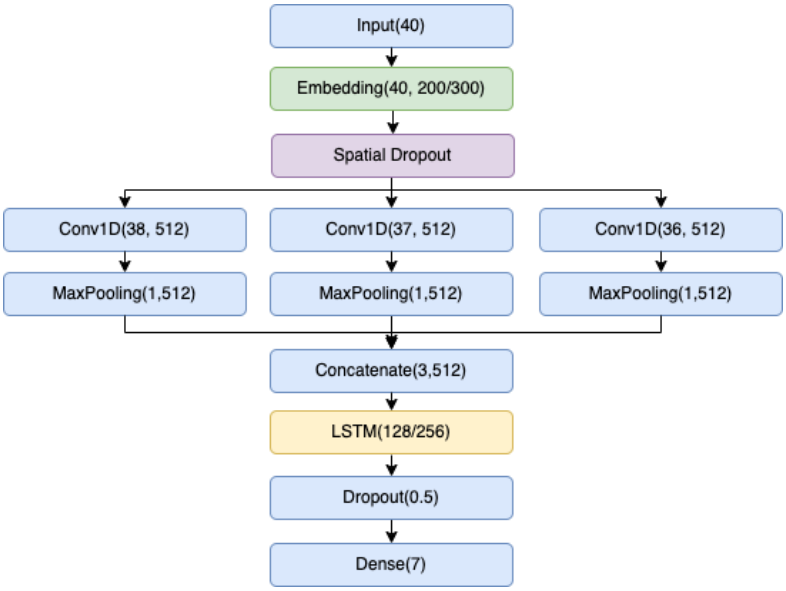


Abbildung 5.7: CNN-LSTM-Modell Architektur

Der größte Unterschied zwischen dem CNN-LSTM-Modell und dem LSTM-CNN-Modell besteht darin, dass die LSTM-Schicht hinter der CNN-Schicht liegt, nämlich zwischen der Concatenate-Schicht und der Dropout-Schicht. Darüber hinaus gibt die LSTM-Schicht wie die LSTM-Schicht des LSTM-Modells nur die letzte Ausgabe zurück. Die Flatten-Schicht wird weggelassen, weil die logische Reihenfolge der Ausgabe der Concatenate-Schicht beibehalten werden soll. Die anderen Schichten des CNN-LSTM-Modells sind die gleichen wie die des LSTM-CNN-Modells.

5.5.2 Resultate

Die Tabelle 5.4 zeigt die Ergebnisse der Experimente der 12 CNN-LSTM-Modellvarianten.

Embedding	Embedding size	LSTM Units	Spatial Dropout	Accuracy(%)	F1 score(%)
GloVe-Tweets	200	128	Yes	57,10	52,42
GloVe-6B	300			57,15	51,03
Word2Vec	300			59,48	54,63
GloVe-Tweets	200		No	59,11	54,00
GloVe-6B	300			55,35	51,99
Word2Vec	300			58,72	53,42
GloVe-Tweets	200	256	Yes	56,82	53,21
GloVe-6B	300			55,97	51,60
Word2Vec	300			59,20	53,87
GloVe-Tweets	200		No	60,14	54,65
GloVe-6B	300			58,59	55,08
Word2Vec	300			61,54	56,10

Tabelle 5.4: Resultate der Experimente der CNN-LSTM-Modellvarianten

Die leistungsstärkste Modellvariante ist das Word2Vec-Modell mit 256 LSTM-Units ohne einer Spatial-Dropout-Schicht mit einem F1-Score von 56,10% und einer Genauigkeit von 61,54%. Eine Spatial-Dropout-Schicht verbesserte nur den F1-Score des Word2Vec-Modells mit 128 LSTM-Units, verschlechterte aber die Genauigkeit des GloVe-Tweets-Modells mit 128 LSTM-Units. Bei Modellen mit 256 LSTM-Units verschlechterten sich die Genauigkeit und der F1-Score durch eine Spatial-Dropout-Schicht.

Mit einer Verdoppelung von LSTM-Units hat sich die Genauigkeit der Modelle mit einer Spatial-Dropout-Schicht verschlechtert. Der F1-Score der GloVe-Modelle ist gestiegen. Allerdings stiegen die Genauigkeit und der F1-Score der Modelle ohne einer Spatial-Dropout-Schicht an, als die Anzahl von LSTM-Units der Modelle verdoppelt wurde.

Wenn nur die Worteinbettung variiert, schnitten die Modelle mit der Word2Vec-Worteinbettung häufig am besten ab. Es gibt jedoch eine Ausnahme. Das beste Modell unter den Modellen mit 128 LSTM-Units ohne einer Spatial-Dropout-Schicht ist das Modell mit der GloVe-Tweets-Worteinbettung.

Abbildung 5.8 zeigt die Veränderung des F1-Scores des besten CNN-LSTM-Modells, nämlich das Word2Vec-Modell mit 256 LSTM-Units ohne einer Spatial-Dropout-Schicht. Dieses Modell erzielte in der 2. Epoche den besten F1-Score des Val-Datensatzes. Der F1-Score des Training-Datensatzes liegt ab der 2. Epoche über 80%. Danach stieg dieser F1-Score immer weiter an. In der 5. Epoche lag er bei etwa 95%. Jedoch hat sich der F1-Score des Val-Datensatzes nach der 2. Epoche verschlechtert. Es gab ein Overfitting beim Training. Dieses Training wurde deshalb nach der 5. Epoche früher beendet.

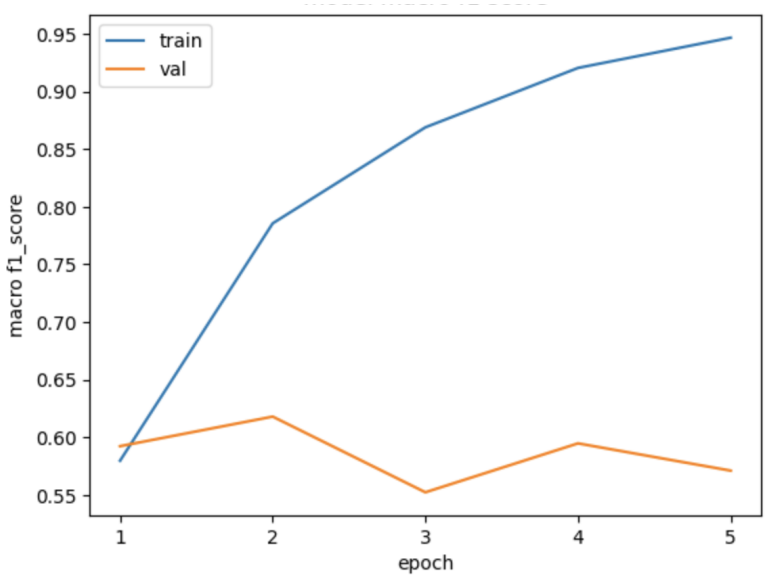


Abbildung 5.8: F1-Score des Word2Vec-CNN-LSTM-Modells mit 256 LSTM-Units ohne Spatial-Dropout

Verglichen mit dem besten CNN-Modell hat dieses Modell einen um etwa 0,1% höheren F1-Score beim Test-Datensatz. Das beste CNN-Modell hat eine Spatial-Dropout-Schicht, während das beste CNN-LSTM-Modell keine Spatial-Dropout-Schicht hat. Darüber hinaus war das Over-fitting beim Training des CNN-LSTM-Modells stärker als beim CNN-Modell.

5.6 Ensemble-Modell

5.6.1 Architektur

Albu und Spinu [10] entwickelten ein Ensemble-Modell zur Erkennung von Emotionen aus Tweets. Das Ensemble-Modell kombiniert ein BERTweet-Modell und ein SVM-Modell, indem dieses Ensemble-Modell die Summe der Wahrscheinlichkeiten der beiden Modelle berechnet.

Das Ensemble-Modell der Arbeit basiert auf dem LSTM-Modell und dem CNN-Modell der Arbeit. Das LSTM-Modell und das CNN-Modell verwenden die gleiche Worteinbettung und berechnen parallel die Wahrscheinlichkeiten der Emotionsklassen. Der Durchschnitt der Wahrscheinlichkeiten pro Klasse der beiden Modelle wird als die Ausgabe des

Ensembles verwendet. Diese Wahrscheinlichkeitsberechnung der Emotionsklassen basiert auf dem Ensemble-Modell von Albu und Spinu. Die Abbildung 5.9 zeigt die Architektur des Ensemble-Modells.

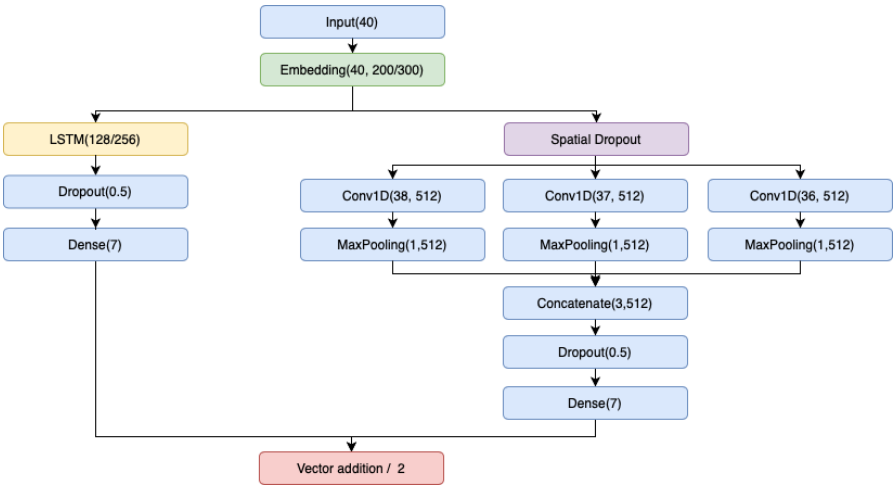


Abbildung 5.9: Ensemble-Modell Architektur

5.6.2 Resultate

Die Resultate der Experimente des Ensemble-Modells werden in der Tabelle 5.5 gezeigt. Das Word2Vec-Modell mit 256 LSTM-Units ohne einer Spatial-Dropout-Schicht ist mit einer Genauigkeit von 61,04% und einem F1-Score von 54,75% das leistungsfähigste. Das GloVe-6B-Modell hat unter den Modellen mit 128 LSTM-Units mit einer Spatial-Dropout-Schicht den höchsten F1-Score. In allen anderen Fällen haben die Modelle mit der Word2Vec-Worteinbettung die beste Leistung. Sowohl eine Spatial-Dropout-Schicht also auch die Verdoppelung von LSTM-Units verbessern nicht immer die Genauigkeit oder den F1-Score.

Embedding	Embedding size	LSTM Units	Spatial Dropout	Accuracy(%)	F1 score(%)
GloVe-Tweets	200	128	Yes	57,12	48,09
GloVe-6B	300			58,63	54,19
Word2Vec	300			60,29	53,50
GloVe-Tweets	200		No	59,83	53,10
GloVe-6B	300			55,77	52,21
Word2Vec	300			59,99	53,11
GloVe-Tweets	200	256	Yes	60,07	52,10
GloVe-6B	300			59,18	53,86
Word2Vec	300			58,46	54,09
GloVe-Tweets	200		No	59,38	53,85
GloVe-6B	300			56,08	52,34
Word2Vec	300			61,04	54,75

Tabelle 5.5: Resultate der Experimente der Ensemble-Modellvarianten

Die Veränderung des F1-Scores des besten Ensembles wird in der folgenden Abbildung 5.10 gezeigt. Dieses Modell erzielt nach der 3. Epoche den höchsten F1-Score des Val-Datensatzes. Der F1-Score des Training-Datensatzes ist immer aufstiegen und liegt ab der 5. Epoche über 90%. Allerdings ist der F1-Score des Val-Datensatzes nach der 3. Epoche stark abgestiegen.

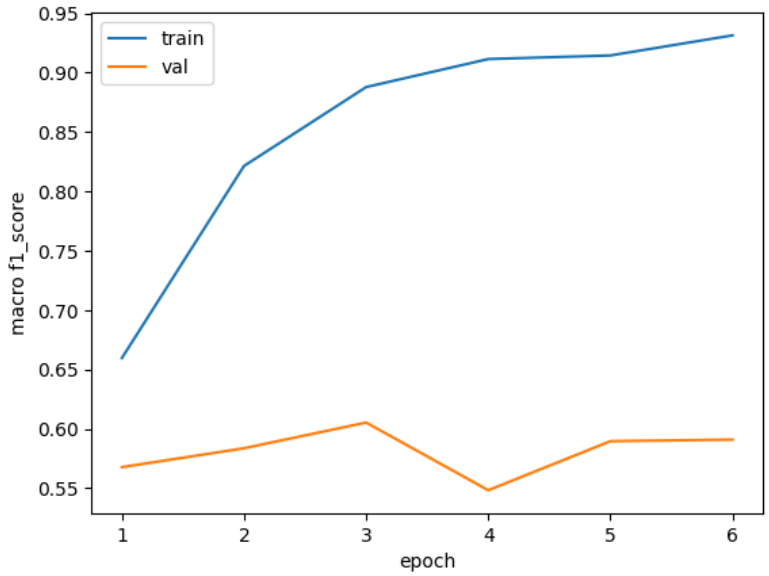


Abbildung 5.10: F1-Score des Word2Vec-Ensemble-Modells mit 256 LSTM-Units ohne Spatial-Dropout

Der F1-Score dieses Ensemble-Modells beim Test-Datensatz ist um 0,1 % niedriger als der des besten LSTM-Modells und um etwa 1% niedriger als der des besten CNN-Modells. Das Ensemble hat die Leistung in diesem Fall nicht verbessert.

6 Resultate im Vergleich und Diskussion

In diesem Kapitel wird aus jedem Modelltyp eine Modellvariante mit dem höchsten F1-Score ausgewählt. Anschließend werden die Ergebnisse dieser Modelle miteinander verglichen. Darüber hinaus werden die Ergebnisse des besten Modells mit LIME untersucht, ob dieses Modell Texte aus logischen Wörtern klassifiziert.

6.1 Ergebnisse im Vergleich

6.1.1 Vergleich mit F1-Score und Genauigkeit

Die folgende Tabelle 6.1 zeigt die Modellvarianten mit dem höchsten F1-Score von jedem Modelltyp. Das LSTM-CNN-Modell hat den höchsten F1-Score von 57,15% und die höchste Genauigkeit von 62,61%. Es ist mit der GloVe-Tweets-Worteinbettung. Dies kann daran liegen, dass die GloVe-Tweets-Worteinbettung auf der Grundlage von Tweets trainiert wurde. Der Training-Datensatz dieser Arbeit besteht aus Reddit-Kommentaren. Tweets und Reddit-Kommentaren sind beide kurze Texte und beinhalten Slang, Kurzformen und Phrasenabkürzungen. Zudem enthält diese Worteinbettung mehr Wörter aus dem Training-Datensatz (siehe 4.2). Die anderen Modelle wurden mit der Word2Vec-Worteinbettung trainiert. Die Vektorgröße der Word2Vec-Worteinbettung ist 300, während die Vektorgröße der GloVe-Tweets-Worteinbettung 200 ist. Dies könnte ein Vorteil für Word2Vec gegenüber GloVe-Tweets sein, da je größer der Vektor ist, desto mehr Merkmale können abgebildet werden.

Modell	Embedding	LSTM Units	Spatial Dropout	Accuracy(%)	F1 score(%)
LSTM	Word2Vec	128	-	60,23	54,86
CNN	Word2Vec	-	Yes	61,91	55,97
LSTM-CNN	GloVe-Tweets	256	No	62,61	57,14
CNN-LSTM	Word2Vec	256	No	61,54	56,10
Ensemble	Word2Vec	256	No	61,04	54,75

Tabelle 6.1: Ergebnisse der beste Modellvariante aus jedem Modelltyp

Das LSTM-CNN-Modell ist das einzige Modell, bei dem die LSTM-Schicht die gesamte Ausgabesequenz zurückgibt. Die anderen LSTM-Schichten geben nur die letzte Ausgabe zurück. Dadurch erhält die Ausgabe der LSTM-Schicht des LSTM-CNN-Modells mehr Informationen. Der Nachteil dabei ist, dass der Rechenaufwand größer ist, weil die Ausgabe der LSTM-Schicht die Eingabe für die nachfolgenden CNN-Schichten ist. Diese CNN-Schichten verarbeiten deshalb 40 Mal mehr Daten, als wenn die LSTM-Schicht nur die letzte Ausgabe zurückliefert.

Das beste LSTM-Modell besteht aus 128 LSTM-Units. Allerdings arbeiten die anderen Modelle mit 256-LSTM-Units. Das CNN-Modell wurde mit Spatial-Dropout trainiert, während die anderen Modelle ohne Spatial-Dropout sind. Eine Verdoppelung der LSTM-Units und ein Spatial-Dropout bringen nicht immer eine bessere Leistung.

6.1.2 Vergleich mit Confusion-Matrix

Um eine detailliertere Aufschlüsselung der Genauigkeit des LSTM-CNN-Modells zu erhalten, wird eine Konfusionsmatrix verwendet. Die linke Seite der Abbildung 6.1 ist eine nicht-normalisierte Konfusionsmatrix, wobei die Summe einer Zeile die Anzahl der Texten einer Emotionsklasse angibt. Da der Test-Datensatz nicht balanciert ist, wird zusätzlich eine normalisierte Konfusionsmatrix verwendet, damit die Genauigkeit pro Klasse eindeutig ist. In einer normalisierten Konfusionsmatrix beträgt die Summe jeder Zeile 100%, was 100 % der Elemente in einer Emotionsklasse darstellt. Diese normalisierte Konfusionsmatrix ist rechts in der Abbildung 6.1 dargestellt. [3]

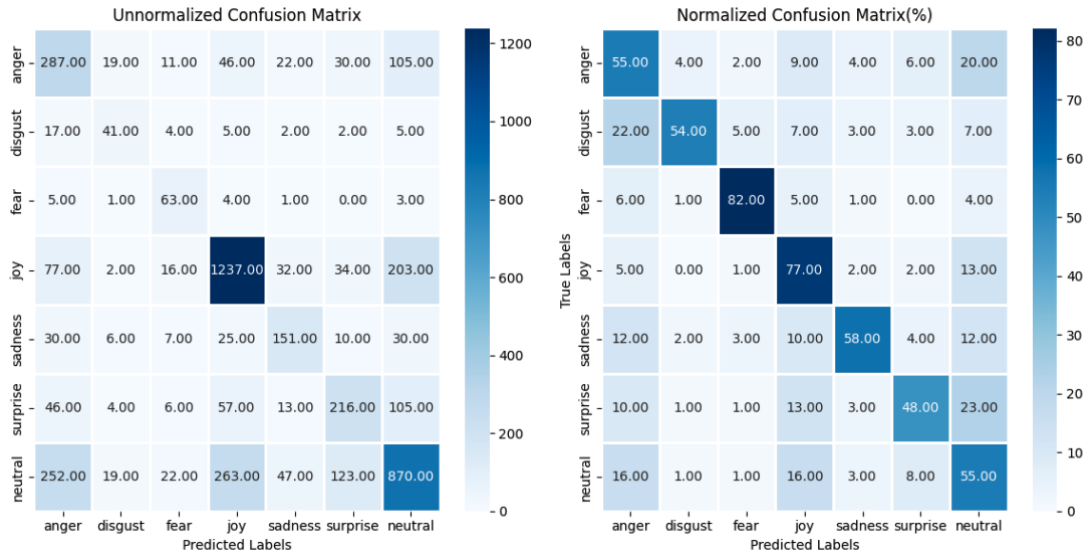


Abbildung 6.1: Konfusionsmatrix für das LSTM-CNN-Modell

Die Emotionsklasse Fear hat die höchste Genauigkeit von 82%. Es folgt die Emotionsklasse Joy mit einer Genauigkeit von 77%. Die Klassen Anger, Disgust, Sadness und Neutral haben jeweils eine Genauigkeit von 54% bis 58%. Die Klasse Surprise hat mit 48% die geringste Genauigkeit.

Bei Betrachtung der falsch klassifizierten Texten kann festgestellt werden, dass die Texte mit dem Label Disgust am häufigsten als Anger klassifiziert wurden und die Texte mit allen anderen Labels außer Fear am häufigsten als Neutral klassifiziert wurden. 23% der Texte mit dem Label Surprise, 20% der Texte mit dem Lable Anger, 13% der Texte mit dem Label Joy und 12% der Texte mit dem Label Sadness wurden als Neutral eingestuft. Außerdem wurden 7% der Texte mit dem Label Disgust als Neutral klassifiziert.

Die Emotionsklasse Neutral hat eine Genauigkeit von 55%. Jeweils 16% der Texte mit dem Label Neutral wurden als Anger oder Joy identifiziert. Für dieses Modell ist es am schwierigsten, die Texte mit dem Label Surprise zu identifizieren. 46% der Texte mit dem Label Surprise wurden als Neutral, Joy oder Anger eingestuft. 13% der Texte mit dem Label Surprise und 16% der Texte mit dem Label Neutral wurden als Joy klassifiziert. Es ist merkwürdig, dass 10% der Texte mit dem Label Sandness als Joy eingestuft wurden, weil die Emotionen von Sandness und Joy gegenteilig sind. Das könnte daran liegen, dass dieses Modell anhand einiger falscher Wörter die Texte als Joy klassifiziert hat. Zudem

wurden 22% der Texte mit dem Label Disgust, 16% der Texte mit dem Label Neutral und 12% Texte mit dem Label Sadness als Anger identifiziert.

Alles in allem ist es beim LSTM-CNN-Modell wichtig, die Merkmale von Anger, Disgust, Sadness, Neutral und Surprise weiter zu lernen, damit das Modell diese Klassen von den anderen Klassen besser unterscheiden kann.

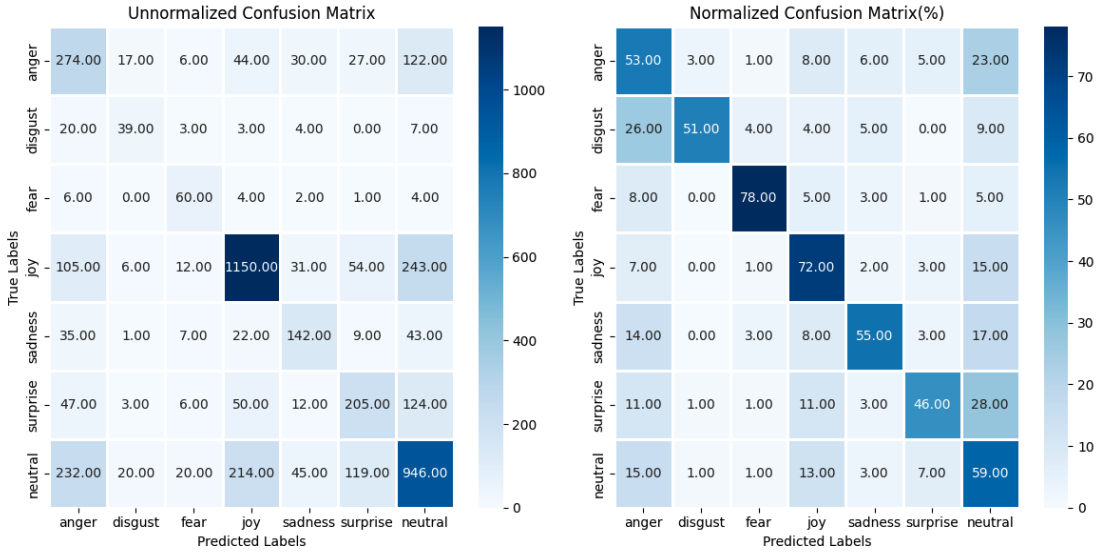


Abbildung 6.2: Konfusionsmatrix für das CNN-LSTM-Modell

Verglichen mit dem LSTM-CNN-Modell weist das CNN-LSTM-Modell jeweils eine um 2% bis 5% niedrigere Genauigkeit bei der Erkennung der Texte mit allen Labels außer Neutral auf. Die Texte mit dem Label Neutal erkannt das CNN-LSTM-Modell jedoch mit einer um 4% höheren Genauigkeit besser als das LSTM-CNN-Modell. Dieses Modell kann die Emotionensklassen Fear und Joy gut klassifizieren, beide mit einer Genauigkeit über 70%. Die Emotion Surprise ist auch die Klasse, die für das CNN-LSTM-Modell am schwierigsten zu identifizierende ist .

Jeweils 1% bis 5% mehr Texte mit allen Labels außer Neutral wurden durch das CNN-LSTM-Modell als Neutral klassifiziert als durch das LSTM-CNN-Modell. Darüber hinaus wurden jeweils 1% bis 4% mehr Texte mit allen Labels außer Anger als Anger eingestuft. Jeweils 1% bis 3% weniger Texte mit allen Labels außer Joy wurden als Joy klassifiziert.

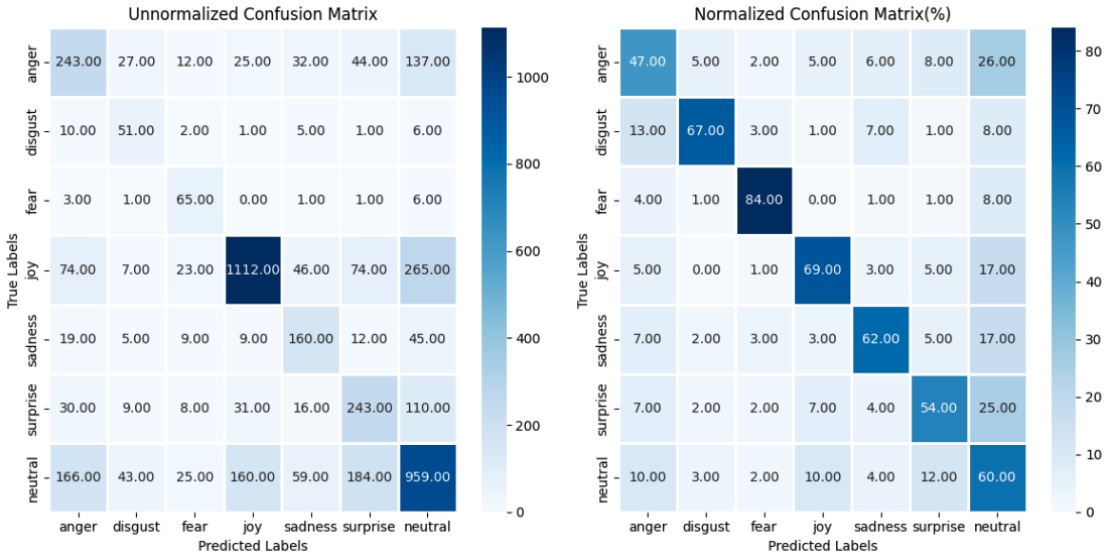


Abbildung 6.3: Konfusionsmatrix für das CNN-Modell

Das CNN-Modell schneidet bei der Klassifizierung der Emotion Anger und Joy deutlich schlechter ab als das LSTM-CNN-Modell, mit jeweils 8% geringerer Genauigkeit. Allerdings erkennt es die andere Emotionen besser. Insbesondere hat es eine 13% höhere Genauigkeit bei der Emotion Disgust.

Jeweils 2% bis 9% weniger Texte mit allen Labels außer Anger und Joy wurden als Anger klassifiziert. Darüber hinaus wurden jeweils 1% bis 6% mehr Text mit allen Labels außer Neutral als Neutral betrachtet. Jeweils 4% bis 7% weniger Texte mit anderen Labels außer Joy wurden als Joy eingestuft.

Die Konfusionsmatrixs des LSTM-Modells und des Ensemble-Modells befinden sich im Anhang A.5. Alle Modelle erkennen die Emotionen Fear und Joy am besten. Während das LSTM-CNN-Modell, das CNN-LSTM-Modell und das LSTM-Modell die Emotion Surprise am schwierigsten erkennen, hat das CNN-Modell und Ensemble-Modell die größten Schwierigkeiten bei der Erkennung der Emotion Anger.

Bei falscher Erkennung werden Texte am häufigsten als Neutral, Joy oder Anger eingestuft. Viele Texte wurden zudem als Neutral falsch eingestuft. Dies kann daran liegen, dass viele Wörter für sich betrachtet neutral sind. Wenn diese Wörter in einem bestimmten Kontext stehen, wird jedoch eine nicht neutrale Bedeutung ausgedrückt.

6.2 Erklärung der Ergebnisse mit LIME

Um ein gesamtes Modell zu bewerten, wird SP-LIME verwendet, damit dieser Bewertungsprozess weniger aufwändig ist. Bei SP-LIME werden zunächst einige unähnliche Texte ausgewählt, sodass mehrere Merkmale beinhaltet werden. Dann werden diese ausgewählten Texte interpretiert, indem angegeben wird, welche Wörter vom Modell zur Klassifizierung dieser Texte verwendet werden.

Hier wird das LSTM-CNN-Modell mit SP-LIME evaluiert. Für die Evaluierung werden 5 Texte aus dem Test-Datensatz ausgewählt. Für die zwei wahrscheinlichsten Emotionsklassen wird dargestellt, anhand welcher Wörter diese Klassifizierung getroffen wird.

Abbildung 6.4: Erklärung für einen Text mit dem Label Joy, der vom Modell als Joy klassifiziert wurde.

Der Text „Lol that's Sammie coates replacement not minds take it how you want to I am back" mit dem Label Joy wurde vom Modell als Joy eingestuft. Die genaue Erklärung ist in der Abbildung 6.4 dargestellt. Nach der Klassifizierung des Modells hat der Text eine 97% Wahrscheinlichkeit für Joy.

Das Wort „lol" steht für „laughing out loud", das meist verwendet wird, um ein Lächeln oder eine leichte Belustigung anzuzeigen [61]. Dieses Wort führt zu einer 80% Wahrscheinlichkeit der Emotionsklasse Joy, was sinnvoll ist. Die Wörter: „replacement", „that", „mind", „to" und „is" sind hingegen neutral, die zusammen eine Wahrscheinlichkeit von 10% für Joy ergeben.

Abbildung 6.5: Erklärung für einen Text mit dem Label Joy, der vom Modell als Joy klassifiziert wurde.

Die Abbildung 6.5 erklärt die Klassifizierung des Textes „So more? No thanks. Love the guys, but that's for side projects.“ mit dem Label Joy. Der Text wurde vom Modell anhand der Wörter „love“ und „thanks“ als Joy klassifiziert. Das Wort „love“ hat im Kontext „love the guy“ die Emotion Joy. Allerdings ist das Wort „thanks“ im Kontext „no thanks“ neutral.

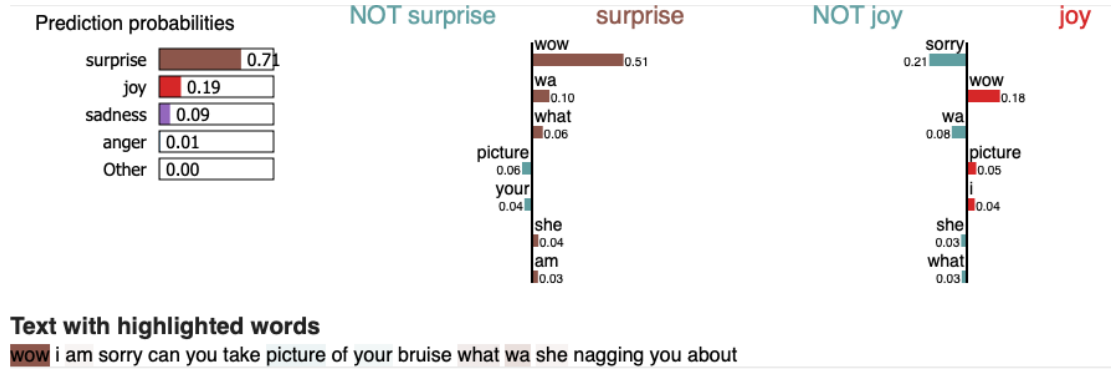


Abbildung 6.6: Erklärung für einen Text mit dem Label Neutral, der vom Modell als Surprise klassifiziert wurde.

Die Erklärung der Klassifizierung des Texts „Wow I'm sorry. Can you take pictures of your bruises? What was she nagging you about?“ wird in der Abbildung 6.6 dargestellt. Dieser Text gehört zur Emotionsklasse Neutral, wurde jedoch vom Modell als Surprise klassifiziert. Diese Klassifizierung basierte auf den Wörtern „wow“, „what“, „was“, „she“ und „am“. Durch das Wort „wow“ wird häufig die Emotion Surprise ausgedrückt. Das Wort „what“ kann in machen Kontexten ebenfalls die Emotion Surprise ausdrücken. Die

anderen Wörter „was“, „she“ und „am“ sind jedoch neutral. Darüber hinaus ist der gemeinsame Kontext Neutral.

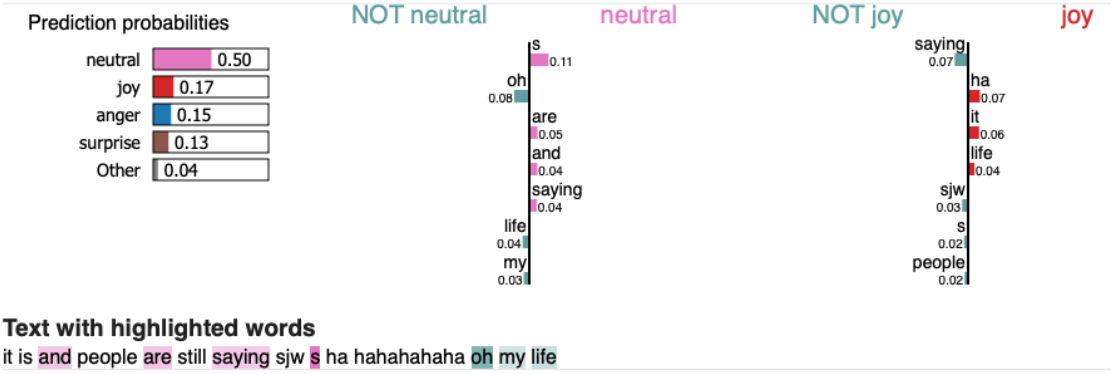


Abbildung 6.7: Erklärung für einen Text mit dem Label Joy, der vom Modell als Neutral klassifiziert wurde.

In der Abbildung 6.7 wird die Erklärung der Klassifizierung des Texts „It’s 2019 and people are *still* saying SSJW’s?"Ha! Hahahahaha! Oh my life.“ dargestellt. Dieser Text mit dem Label Joy wurde vom Modell als Neutral eingestuft. Die Wörter „s“, „are“, „and“ und „saying“ sind neutral. Allerdings hat das Modell nicht erkannt, dass die Wörter „ha“ und „hahahahaha“ nicht neutral sind. Aufgrund des Wortes „ha“ hat der Text jedoch eine Wahrscheinlichkeit von 7%, zur Klasse Joy zu gehören.

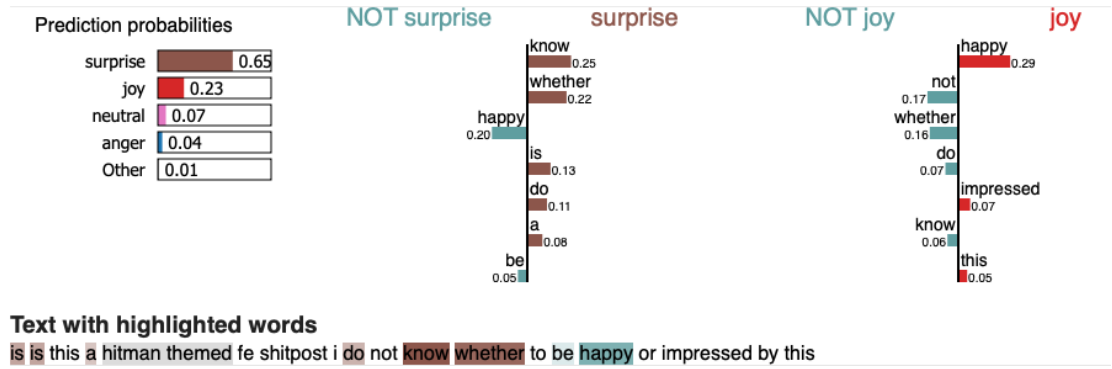


Abbildung 6.8: Erklärung für einen Text mit dem Label Joy, der vom Modell als Surprise klassifiziert wurde.

Die Erklärung für den Text „Is...Is this a Hitman themed FE Shitpost?...I dunno whether to be happy or impressed by this.“ ist in der Abbildung 6.8 beschrieben. Der Text hat das Label Joy. Das Modell stufte ihn auf Grundlage der Wörter „know“, „whether“, „is“, „do“ und „a“ als Surprise ein. Diese Wörter sind jedoch neutral. Das Modell erkannte, dass

die Wörter „happy“ und „impressed“ die Emotion Joy ausdrücken. Allerdings ergeben die jeweils nur eine Wahrscheinlichkeit von 29% und 7% für Joy.

Für die Klassifizierung der Emotionen ist es sehr wichtig, den gesamten Textkontext zu verstehen. Es scheint, dass es für das Modell manchmal schwierig ist, die Bedeutung eines ganzen Texts zu verstehen. Außerdem ist es wichtig, dass das Modell weiterhin die Wichtigkeit eines Wort in einem bestimmten Kontext lernt. In den letzten beiden Beispielen hat es den Zusammenhang zwischen Wörtern und der Emotionen bereits teilweise erkannt. Aber es erkannte manchmal nicht, wie wichtig diese Wörter in dem Kontext sind. Beispielsweise sollten die Wörter „ha“ und „hahahahaha“ eine höhere Wahrscheinlichkeit von Joy ergeben.

Hinzu kommt, dass das Modell zwar manchmal einen Text richtig klassifiziert, aber die Wörter, die zur Klassifizierung führen, nicht alle sinnvoll sind. Zum Beispiel sollten die Wörter „your“ und „my“ keinen Einfluss auf die Klassifizierung haben. Aus den Erläuterungen zu den Texten geht jedoch hervor, dass diese beiden Wörter zu Joy beitragen.

Bei der Erläuterung der anderen Modelle mit SP-LIME (siehe Anhang A.4) kann festgestellt werden, dass die anderen Modelle ähnliche Probleme wie das LSTM-CNN-Modell haben: sie haben Schwierigkeiten, den gesamten Textkontext zu verstehen und haben manchmal Texte anhand nicht sinnvollen Wörtern klassifiziert.

7 Fazit

7.1 Zusammenfassung

Um Texte nach Emotionen zu klassifizieren, wurden in der Arbeit verschiedene neuronale Netze untersucht und bewertet, nämlich das LSTM-Modell, das CNN-Modell, das CNN-LSTM-Modell, das LSTM-CNN-Modell und das Ensemble-Modell. Die Leistungen dieser Modelle wurden anhand der Genauigkeit und des F1-Scores miteinander verglichen. Darüber hinaus wurden die Balancierung eines Datensatzes und die Textvorverarbeitung vorgestellt. Drei verschiedene Worteinbettungen wurden für jedes Modell verwendet und verglichen. Des Weiteren wurden die Einflüsse der Anzahl der LSTM-Units auf die Leistung der Modelle untersucht. Außerdem wurde getestet, ob die Leistung der Modelle durch eine Spatial-Dropout-Schicht verbessert wird.

Das leistungsfähigste Modell ist das LSTM-CNN-Modell mit einer Genauigkeit von 62,61% und einem F1-Score von 57,14%. Es folgt das CNN-LSTM-Modell mit einer Genauigkeit von 61,54% und einem F1-Score von 56,10%. Das CNN-Modell hat eine besser Leistung als das LSTM-Modell. Das Ensemble-Modell erzielte den niedrigsten F1-Score von 54,75%.

Die Modellen erzielen am häufigsten die beste Ergebnisse mit der Word2Vec-Worteinbettung. Das beste LSTM-CNN-Modell hat jedoch die GloVe-Tweets-Worteinbettung.

In den Experimenten wurden Modelle mit 128 LSTM-Units und 256 LSTM-Units getestet. Durch Experimente mit den Modellvarianten wird festgestellt, dass ein Modell mit einer Verdoppelung der Anzahl LSTM-Units nicht immer eine bessere Leistung erreicht. Mit einer Spatial-Dropout-Schicht hat das CNN-Modell sich immer verbessert. Die anderen Modelle verbesserten sich damit nicht immer. Während das beste CNN-Modell mit einer Spatial-Dropout-Schicht ist, sind die besten Modellen der anderen Typen ohne einer Spatial-Dropout-Schicht. Das beste LSTM-Modell ist mit 128 LSTM-Units, während die andren Modellen mit 256 LSTM-Units die beste Leistung erreicht haben.

Die Konfusionmatrix wurde verwendet, um die Genauigkeit der Klassifizierungen der besten Modelle aus jedem Modelltyp pro Klasse zu betrachten. Die Klassen Fear und Joy haben bei allen Modellen eine Genauigkeit von jeweils ungefähr 70%. Alle Modelle haben Schwierigkeiten, die Klasse Surprise oder Anger zu erkennen. Das CNN-LSTM-Modell und das LSTM-CNN-Modell haben bei der Erkennung der Klasse Disgust eine um etwa 10% geringere Genauigkeit als die anderen Modelle.

Manchmal hat ein Modell eine höhere Genauigkeit für eine Klasse. Mehrere Texte der anderen Klassen werden fälschlicherweise als diese Klasse klassifiziert. Dies kann daran liegen, dass bei diesem Modell mehrere Merkmale mit dieser Klasse verbunden sind, aber diese Verbindungen nicht wirklich sinnvoll sind. Das CNN-Modell hat beispielsweise eine höhere Genauigkeit bei Surprise als andere Modelle. Mehrere Texte mit anderen Labels wurden jedoch als Surprise klassifiziert.

Die meisten Texte werden fälschlicherweise als Neutral klassifiziert. Der Grund dafür könnte sein, dass die Modelle einen Text anhand einzelner Wörter klassifizieren. Da viele Wörter neutral sind, ist die Wahrscheinlichkeit für die Klasse Neutral höher als für die richtige Klasse. Darüber hinaus werden viele Texte falsch als Joy oder Anger erkannt. Es ist wichtig für das LSTM-CNN Modell, weiterhin die Merkmale der Klasse Anger, Disgust, Sadness, Surprise und Neutral zu lernen.

Die Klassifizierungen des LSTM-CNN-Modells wurden zusätzlich mit LIME erklärt. Das Modell hat gelernt, welche Wörter mit welcher Emotion verbunden sind. Allerdings sind die gelernten Verbindungen zwischen Wörtern und Emotionen nicht immer richtig. Es ist eine Herausforderung für das Modell, den Textkontext zu erkennen. Dies führt auch manchmal zu falschen Ergebnissen. Die anderen Modelle haben die gleichen Probleme.

7.2 Ausblick

In diesem Kapitel werden mögliche Erweiterungen und weiterführend Überlegungen gegeben.

Stop Words
In der Arbeit werden Stop Words beibehalten. Stop Words sind Wörter, die nicht dazu beitragen, den Kontext oder die Bedeutung eines Texts zu verstehen. Je nach dem analysierten Problem und dem verwendeten Datensatz sind Stop Words unterschiedlich [66]. Nach der Erklärung von LIME klassifiziert das beste LSTM-CNN-Modell aufgrund

Wörter wie „I“, „you“, „she“ usw., die in den Fällen Stop Words sind. Während der Textvorverarbeitung können Stop Words gelöscht werden. Dann kann untersucht werden, ob ein Modell mit einem Datensatz ohne Stop Words eine bessere Leistung erreichen kann. Die Schwierigkeit besteht darin, dass für den Datensatz eine List von Stop Words erstellt werden muss, durch welche nicht alle Stop Words entfernt werden. Dies liegt daran, dass die Erkennung der Emotionen zu der Stimmungsanalysis gehört, die sensitive zu der Löschung von Stop Words ist.

Datensatz
Alle Experimente beruhen auf einem Datensatz. Die Ergebnisse der Modelle sind nicht allgemein gültig. Diese Experimente können mit einem anderen Datensatz durchgeführt werden. So können die Schlussfolgerungen auf ihre Verallgemeinerbarkeit hin überprüft werden.

Balancierung des Training-Datensatzes
Beim Balancierung des Training-Datensatzes wurden Elemente einer Minderheitsklasse nach dem Zufallsprinzip kopiert. Dies könnte einen Einfluss auf die Qualität des Datensatzes haben. Anhand der Analyse mit der Konfusion-Matrix sind die am schwierigsten zu erkennenden Klassen Surprise oder Anger, die beides zu den Minderheitsklassen gehören. Jeweils ungefähr 470 und 120 Texte der beiden Klassen wurden kopiert. Allerdings sind die Texte aus den Klassen Fear und Disgust am meisten kopiert, mit ca. 3500 kopierten Texten pro Klasse, wobei die Klasse Fear für alle Modelle am besten zu erkennen ist. Das Kopieren nach dem Zufallsprinzip führt daher nicht unbedingt zu einer schlechteren Qualität einer Minderheitsklasse.

Für eine bessere Qualität des Datensatzes kann das Over-Sampling manipuliert werden. Beispielsweise können die kopierten Texte der Minderheitsklasse umformuliert werden.

Durchführung der Experiment
Die Leistung jeder Modellvariante wird aus dem besten Training entnommen. Wenn eine Modellvariante eine große Abweichung verglichen mit ähnlichen Modellen hatte, wurde das Training dieser Modellvariante mehrmals durchgeführt. Andernfalls wurde eine Modellvariante nur einmal trainiert. Die Zufälligkeit des Trainings eines neuronalen Netzes hat Auswirkungen auf die Leistung eines Modells. Daher ist es besser, wenn eine durchschnittliche Leistung jeder Modellvariante aus mehreren Trainingsdaten berechnet wird. Dies verursacht jedoch einen höheren Rechenaufwand.

Erkennung der Kontext und der Merkmale der Emotionklassen
Wie in den Abschnitten 6.2 und 7.1 erwähnt und im Anhang A.4 gezeigt, haben diese Modelle manchmal Schwierigkeiten, den Textkontext zu verstehen.

Zudem ist es für diese Modelle besonders schwierig, die Emotionsklasse Surprise zu erkennen. Ein Grund dafür kann sein, dass Fragezeichen manchmal helfen, Surprise auszudrücken. In dieser Arbeit wurden jedoch alle Satzzeichen aus den Texte gelöscht. Beispielsweise gibt es ein Text mit dem Label Surprise im Datensatz: "Well i know this much is true, there's unspoken truth amoung us men, we secretly love hairy vaj. No? Just me?". Der Text nach der Textvorverarbeitung lautet: "well i know this much is true there is unspoken truth amoung us men we secretly love hairy vaj no just me". Die Emotion Surprise wird vor der Textvorverarbeitung durch die beiden Fragezeichen deutlicher ausgedrückt. Es könnte bei der Erkennung von Surprise helfen, Fragezeichen beizubehalten. Darüber hinaus haben diese Modelle auch Schwierigkeiten, die Emotionsklassen Anger, Disgust, Sadness und Neutral zu erkennen. Es ist eine Forschungsrichtung, diese Probleme zu lösen.

Literaturverzeichnis

[1] : *What is the difference between word2Vec and Glove ?* Oktober 2020. – URL https://machinelearninginterview.com/topics/natural-language-processing/what-is-the-difference-between-word2vec-and-glove. – Zugriffsdatum: 2023-07-07

[2] : *F-1 Score for Multi-Class Classification.* November 2022. – URL https://www.baeldung.com/cs/multi-class-f1-score. – Zugriffsdatum: 2023-09-07

[3] : *Confusion Matrix - an overview ScienceDirect Topics.* August 2023. – URL https://www.sciencedirect.com/topics/engineering/confusion-matrix. – Zugriffsdatum: 2023-09-17

[4] : *F1 Score in Machine Learning: Intro & Calculation.* September 2023. – URL https://www.v7labs.com/blog/f1-score-guide. – Zugriffsdatum: 2023-09-07

[5] : *Google Code Archive - Long-term storage for Google Code Project Hosting.* Juni 2023. – URL https://code.google.com/archive/p/word2vec. – Zugriffsdatum: 2023-06-27

[6] : *Was sind neuronale Faltungsnetze (Convolutional Neural Networks)?* Mai 2023. – URL https://www.ibm.com/de-de/topics/convolutional-neural-networks. – Zugriffsdatum: 2023-06-29

[7] : *What are Neural Networks?* Mai 2023. – URL https://www.ibm.com/topics/neural-networks. – Zugriffsdatum: 2023-07-28

[8] : *What is a Neural Network?* September 2023. – URL https://www.databricks.com/glossary/neural-network. – Zugriffsdatum: 2023-07-28

[9] Acheampong, Francisca A. ; Wenyu, Chen ; Nunoo-Mensah, Henry: Text-based emotion detection: Advances, challenges, and opportunities. In: *Engineering Reports*

2 (2020), Nr. 7, S. e12189. – URL https://onlinelibrary.wiley.com/doi/abs/10.1002/eng2.12189. – Zugriffsdatum: 2023-07-07. – ISSN 2577-8196

[10] Albu, Ionuţ-Alexandru ; Spînu, Stelian: Emotion Detection From Tweets Using a BERT and SVM Ensemble Model URL http://arxiv.org/abs/2208.04547, August 2022. – Zugriffsdatum: 2023-07-25. – Forschungsbericht. arXiv:2208.04547 [cs] type: article

[11] Alshubaily, Ibrahim: TextCNN with Attention for Text Classification. URL http://arxiv.org/abs/2108.01921, August 2021. – Zugriffsdatum: 2023-06-20. – Forschungsbericht. arXiv:2108.01921 [cs] type: article

[12] Awati, Rahul: convolutional neural network (CNN). In: *Enterprise AI* (2023), April. – URL https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network. – Zugriffsdatum: 2023-07-29

[13] Baheti, Pragati: *Activation Functions in Neural Networks [12 Types and Use Cases]*. Mai 2023. – URL https://www.v7labs.com/blog/neural-networks-activation-functions. – Zugriffsdatum: 2023-07-28

[14] Balaji, M. ; Yuvaraj, N.: Intelligent Chatbot Model to Enhance the Emotion Detection in social media using Bi-directional Recurrent Neural Network. In: *Journal of Physics: Conference Series* 1362 (2019), nov, Nr. 1, S. 012039

[15] Brownlee, Jason: *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. Juli 2017. – URL https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/. – Zugriffsdatum: 2023-07-01

[16] Brownlee, Jason: What Are Word Embeddings for Text? - MachineLearningMastery.com. In: *MachineLearningMastery* (2019), August. – URL https://machinelearningmastery.com/what-are-word-embeddings. – Zugriffsdatum: 2023-07-09

[17] Brownlee, Jason: How Do Convolutional Layers Work in Deep Learning Neural Networks? In: *MachineLearningMastery* (2020), April. – URL https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks. – Zugriffsdatum: 2023-07-09

[18] Brownlee, Jason: Why One-Hot Encode Data in Machine Learning? - MachineLearningMastery.com. In: *MachineLearningMastery* (2020), Juni.

– URL https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning. – Zugriffsdatum: 2023-07-10

[19] Brownlee, Jason: Difference Between a Batch and an Epoch in a Neural Network - MachineLearningMastery.com. In: *MachineLearningMastery* (2022), August. – URL https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch. – Zugriffsdatum: 2023-07-10

[20] Cahyani, Denis E. ; Wibawa, Aji P. ; Prasetya, Didik D. ; Gumilar, Langlang ; Akhbar, Fadhilah ; Triyulinar, Egi R.: Text-Based Emotion Detection using CNN-BiLSTM. In: *2022 4th International Conference on Cybernetics and Intelligent System (ICORIS)*, Oktober 2022, S. 1–5

[21] Delua, Julianna: *Supervised vs. Unsupervised Learning: What's the Difference?* November 2022. – URL https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning. – Zugriffsdatum: 2023-06-17

[22] Flomo, Gabe: *Text-Preprocessing: Wie bereitet man Text erfolgreiches Machine Learning vor?* Januar 2019. – URL https://entwickler.de/machine-learning/text-preprocessing-wie-bereitet-man-text-fur-erfolgreiches-machine-learning-vor. – Zugriffsdatum: 2023-07-12

[23] Gal, Yarin ; Ghahramani, Zoubin: A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. URL http://arxiv.org/abs/1512.05287, Oktober 2016. – Zugriffsdatum: 2023-06-18. – Forschungsbericht. arXiv:1512.05287 [stat] type: article

[24] Gillis, Alexander S.: data splitting. In: *Enterprise AI* (2022), April. – URL https://www.techtarget.com/searchenterpriseai/definition/data-splitting. – Zugriffsdatum: 2023-07-04

[25] Gillis, Alexander S.: lemmatization. In: *Enterprise AI* (2023), März. – URL https://www.techtarget.com/searchenterpriseai/definition/lemmatization. – Zugriffsdatum: 2023-07-07

[26] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning*. MIT Press, 2016. – http://www.deeplearningbook.org. – Zugriffsdatum: 2023-07-20

[27] Grandini, Margherita ; Bagli, Enrico ; Visani, Giorgio: Metrics for Multi-Class Classification: an Overview. URL http://arxiv.org/abs/2008.05756, August 2020. – Zugriffsdatum: 2023-06-20. – Forschungsbericht. arXiv:2008.05756 [cs, stat] type: article

[28] Guymonahan: Basic Overviews on Convolutional Neural Networks - Guymonahan - Medium. In: *Medium* (2022), Januar. – URL https://guymonahan.medium.com/basic-overviews-on-convolutional-neural-networks-f205180116be. – Zugriffsdatum: 2023-07-05

[29] Hassler, M. ; Fliedl, G.: Text preparation through extended tokenization. In: *Data Mining VII: Data, Text and Web Mining and their Business Applications*, WIT Press, jun 2006

[30] Hochreiter, Sepp ; Schmidhuber, Jürgen: Long Short-term Memory. In: *Neural Comput.* 9 (1997), Dezember, Nr. 8, S. 1735–80. – ISSN 0899-7667

[31] Holzinger, Andreas: From Machine Learning to Explainable AI. In: *2018 World Symposium on Digital Intelligence for Systems and Machines (DISA)*, IEEE, aug 2018

[32] Iyer, Abhishek ; Das, Srimit S. ; Teotia, Reva ; Maheshwari, Shishir ; Sharma, Rishi R.: CNN and LSTM based ensemble learning for human emotion recognition using EEG recordings. In: *Multimedia Tools and Applications* 82 (2023), Februar, Nr. 4, S. 4883–4896. – URL https://doi.org/10.1007/s11042-022-12310-7. – Zugriffsdatum: 2023-07-27. – ISSN 1573-7721

[33] Jeffrey Pennington, Christopher D. M.: *GloVe: Global Vectors for Word Representation.* April 2018. – URL https://nlp.stanford.edu/pubs/glove.pdf. – Zugriffsdatum: 2023-07-18

[34] Joshi, Saurav ; Jain, Tanuj ; Nair, Nidhi: Emotion Based Music Recommendation System Using LSTM - CNN Architecture. In: *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Juli 2021, S. 01–06

[35] Kim, Yoon: Convolutional Neural Networks for Sentence Classification. URL http://arxiv.org/abs/1408.5882, September 2014. – Zugriffsdatum: 2023-06-20. – Forschungsbericht. arXiv:1408.5882 [cs] type: article

[36] Kingma, Diederik P. ; Ba, Jimmy: Adam: A Method for Stochastic Optimization. URL http://arxiv.org/abs/1412.6980, Januar 2017. – Zugriffsdatum: 2023-07-01. – Forschungsbericht. arXiv:1412.6980 [cs] type: article

[37] Lahera, German: Unbalanced Datasets and What To Do About Them. In: *Medium* (2022), März. – URL https://medium.com/strands-tech-corner/unbalanced-datasets-what-to-do-144e0552d9cd. – Zugriffsdatum: 2023-07-08 – ISSN 1440-5529

[38] Mathews, Sherin M.: Explainable Artificial Intelligence Applications in NLP, Biomedical, and Malware Classification: A Literature Review. In: Arai, Kohei (Hrsg.) ; Bhatia, Rahul (Hrsg.) ; Kapoor, Supriya (Hrsg.): *Intelligent Computing*. Cham : Springer International Publishing, 2019 (Advances in Intelligent Systems and Computing), S. 1269–1292. – ISBN 9783030228682

[39] Mazumder, Saikat: 5 Techniques to Handle Imbalanced Data For a Classification Problem. In: *Analytics Vidhya* (2022), Dezember. – URL https://www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanced-data-for-a-classification-problem. – Zugriffsdatum: 2023-07-15

[40] Mikolov, Tomas ; Chen, Kai ; Corrado, Greg ; Dean, Jeffrey: Efficient Estimation of Word Representations in Vector Space. URL http://arxiv.org/abs/1301.3781. – Zugriffsdatum: 2023-05-19, September 2013. – Forschungsbericht. arXiv:1301.3781 [cs] type: article

[41] Murthy, Ashritha R. ; Kumar, K M A.: A Review of Different Approaches for Detecting Emotion from Text. In: *IOP Conference Series: Materials Science and Engineering* 1110 (2021), mar, Nr. 1, S. 012009

[42] Musstafa: Optimizers in Deep Learning - MLearning.ai - Medium. In: *Medium* (2022), Februar. – URL https://medium.com/mlearning-ai/optimizers-in-deep-learning-7bf81fed78a0. – Zugriffsdatum: 2023-07-07

[43] Olah, Christopher: *Understanding LSTM Networks – colah's blog*. September 2022. – URL https://colah.github.io/posts/2015-08-Understanding-LSTMs. – Zugriffsdatum: 2023-06-30

[44] Pai, Aravindpai: What is Tokenization in NLP? Here's All You Need To Know. In: *Analytics Vidhya* (2023), Mai. – URL https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp. – Zugriffsdatum: 2023-07-03

[45] Pasupa, Kitsuchart ; Seneewong Na Ayutthaya, Thititorn: Thai sentiment analysis with deep learning techniques: A comparative study based on word embedding, POS-tag, and sentic features. In: *Sustainable Cities and Society* 50 (2019), Oktober, S. 101615. – URL https://www.sciencedirect.com/science/article/pii/S2210670718325228. – Zugriffsdatum: 2023-07-18. – ISSN 2210-6707

[46] Pophale, Shraddha ; Gandhi, Hetal ; Gupta, Anil K.: Emotion Recognition Using Chatbot System. In: *Proceedings of International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications*. Springer Singapore, oct 2020, S. 579–587

[47] Rai, Arun: Explainable AI: from black box to glass box. In: *Journal of the Academy of Marketing Science* 48 (2019), dec, Nr. 1, S. 137–141

[48] research google: *GoEmotions-data*. Mai 2020. – URL https://github.com/google-research/google-research/tree/master/goemotions/data. – Zugriffsdatum: 2023-06-02

[49] Ribeiro, Marco T. ; Singh, Sameer ; Guestrin, Carlos: "Why Should I Trust You?": Explaining the Predictions of Any Classifier. URL http://arxiv.org/abs/1602.04938, August 2016. – Zugriffsdatum: 2023-07-07. – Forschungsbericht. arXiv:1602.04938 [cs, stat] type: article

[50] Saeed, Mehreen: *A Guide to Text Preprocessing Techniques for NLP - Blog Scale Virtual Events*. Mai 2023. – URL https://exchange.scale.com/public/blogs/preprocessing-techniques-in-nlp-a-guide. – Zugriffsdatum: 2023-07-12

[51] Samek, Wojciech ; Montavon, Grégoire ; Vedaldi, Andrea ; Hansen, Lars K. ; Müller, Klaus-Robert: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer Nature, September 2019. – Google-Books-ID: j5yuDwAAQBAJ. – ISBN 9783030289546

[52] Saumyab271: Stemming vs Lemmatization in NLP: Must-Know Differences. In: *Analytics Vidhya* (2023), April. – URL https://www.analyticsvidhya.

com/blog/2022/06/stemming-vs-lemmatization-in-nlp-must-know-differences. – Zugriffsdatum: 2023-07-02

[53] SAVYAKHOSLA: *CNN Introduction to Pooling Layer*. April 2023. – URL https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer. – Zugriffsdatum: 2023-06-30

[54] SEYEDITABARI, Armin ; TABARI, Narges ; ZADROZNY, Wlodek: Emotion Detection in Text: a Review. URL http://arxiv.org/abs/1806.00674, Juni 2018. – Zugriffsdatum: 2023-07-01. – Forschungsbericht. – 1 S. arXiv:1806.00674 [cs] type: article

[55] SHAABAN, Yasmin ; KORASHY, Hoda ; MEDHAT, Walaa: Emotion Detection using Deep Learning. In: *2021 16th International Conference on Computer Engineering and Systems (ICCES)*, Dezember 2021, S. 1–10

[56] SHRIVASTAVA, Kush ; KUMAR, Shishir ; JAIN, Deepak K.: An effective approach for emotion detection in multimedia text data using sequence based convolutional neural network. In: *Multimedia Tools and Applications* 78 (2019), Oktober, Nr. 20, S. 29607–29639. – URL https://doi.org/10.1007/s11042-019-07813-9. – Zugriffsdatum: 2023-07-18. – ISSN 1573-7721

[57] SRIVASTAVA, Nitish ; HINTON, Geoffrey ; KRIZHEVSKY, Alex ; SUTSKEVER, Ilya ; SALAKHUTDINOV, Ruslan: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. In: *Journal of Machine Learning Research* 15 (2014), Nr. 56, S. 1929–1958. – URL http://jmlr.org/papers/v15/srivastava14a.html. – Zugriffsdatum: 2023-10-02. – ISSN 1533-7928

[58] STANFORDNLP: *GloVe*. Juni 2023. – URL https://github.com/stanfordnlp/GloVe. – Zugriffsdatum: 2023-06-27

[59] STEPHEN ALLWRIGHT: F1 score vs accuracy, which is the best metric? In: *Stephen Allwright* (2022), August. – URL https://stephenallwright.com/f1-score-vs-accuracy. – Zugriffsdatum: 2023-08-02

[60] TEAM, Keras: *Keras documentation: SpatialDropout1D layer*. Juli 2023. – URL https://keras.io/api/layers/regularization_layers/spatial_dropout1d. – Zugriffsdatum: 2023-07-15

[61] Ticak, Marko: What Does Lol Mean? In: *What Does Lol Mean? | Grammarly* (2019), Mai. – URL https://www.grammarly.com/blog/lol-meaning. – Zugriffsdatum: 2023-09-07

[62] Tompson, Jonathan ; Goroshin, Ross ; Jain, Arjun ; LeCun, Yann ; Bregler, Christopher: Efficient Object Localization Using Convolutional Networks. URL http://arxiv.org/abs/1411.4280, Juni 2015. – Zugriffsdatum: 2023-06-18. – Forschungsbericht. arXiv:1411.4280 [cs] type: article

[63] Tripathi, Ashutosh: *What is the main difference between RNN and LSTM.* Juli 2022. – URL https://ashutoshtripathi.com/2021/07/02/what-is-the-main-difference-between-rnn-and-lstm-nlp-rnn-vs-lstm. – Zugriffsdatum: 2023-06-01

[64] Twin, Alexandra: *Understanding Overfitting and How to Prevent It.* Juni 2023. – URL https://www.investopedia.com/terms/o/overfitting.asp. – Zugriffsdatum: 2023-06-17

[65] Wang, Wenbo ; Chen, Lu ; Thirunarayan, Krishnaprasad ; Sheth, Amit P.: Harnessing Twitter "Big Data"for Automatic Emotion Identification. In: *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, IEEE, sep 2012

[66] Wilame: Why is removing stop words not always a good idea - Wilame - Medium. In: *Medium* (2023), März. – URL https://medium.com/@wila.me/why-is-removing-stop-words-not-always-a-good-idea-c8d35bd77214. – Zugriffsdatum: 2023-08-02 – ISSN 8357-7214

[67] Yadav, Harsh: Dropout in Neural Networks - Towards Data Science. In: *Medium* (2023), Mai. – URL https://towardsdatascience.com/dropout-in-neural-networks-47a162d621d9. – Zugriffsdatum: 2023-07-12

[68] Yse, Diego L.: Text Normalization for Natural Language Processing (NLP). In: *Medium* (2021), Dezember. – URL https://towardsdatascience.com/text-normalization-for-natural-language-processing-nlp-70a314bfa646. – Zugriffsdatum: 2023-07-03 – ISSN 7031-4646

[69] Zhou, Qimin ; Wu, Hao: NLP at IEST 2018: BiLSTM-Attention and LSTM-Attention via Soft Voting in Emotion Classification. In: *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media*

Analysis. Brussels, Belgium : Association for Computational Linguistics, Oktober 2018, S. 189–194. – URL https://aclanthology.org/W18-6226. – Zugriffsdatum: 2023-07-18

[70] Zvornicani, Enes: *Relation Between Learning Rate and Batch Size | Baeldung on Computer Science*. März 2023. – URL https://www.baeldung.com/cs/learning-rate-batch-size. – Zugriffsdatum: 2023-06-30

A Anhang

A.1 Version der verwendeten Bibliotheken

Im Folgenden werden die Versionen der wichtigen Bibliotheken aufgelistet, die in der Arbeit verwendet wurden:

1. numpy==1.23.5
2. keras==2.12.0
3. tensorflow==2.12.0
4. tensorflow-addons==0.21.0
5. pandas==1.5.3
6. matplotlib==3.7.1
7. seaborn==0.12.2
8. nltk==3.8.1
9. sklearn==1.2.2
10. gensim==4.3.1
11. lime==0.2.0.1

A.2 Visualisierung der Architektur der Modelle

Für das CNN-, LSTM-CNN-, CNN-LSTM- und das Ensemble-Modell werden nur die Architekturen mit einer Spatial-Dropout-Schicht gezeigt.

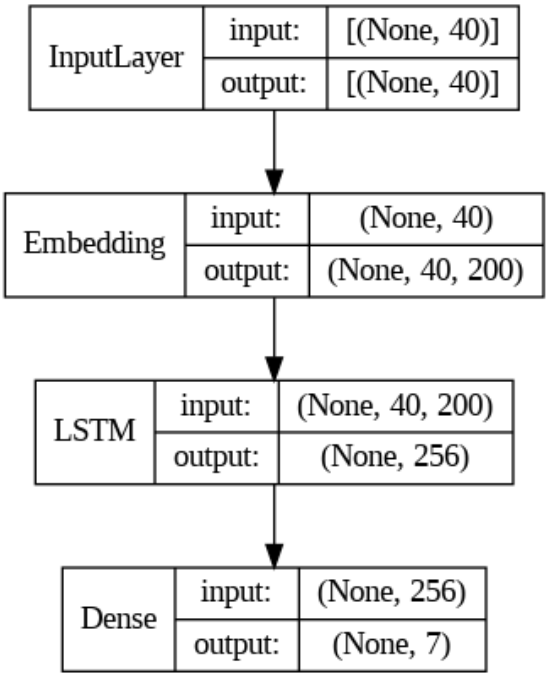


Abbildung A.1: Architektur des LSTM-Modells

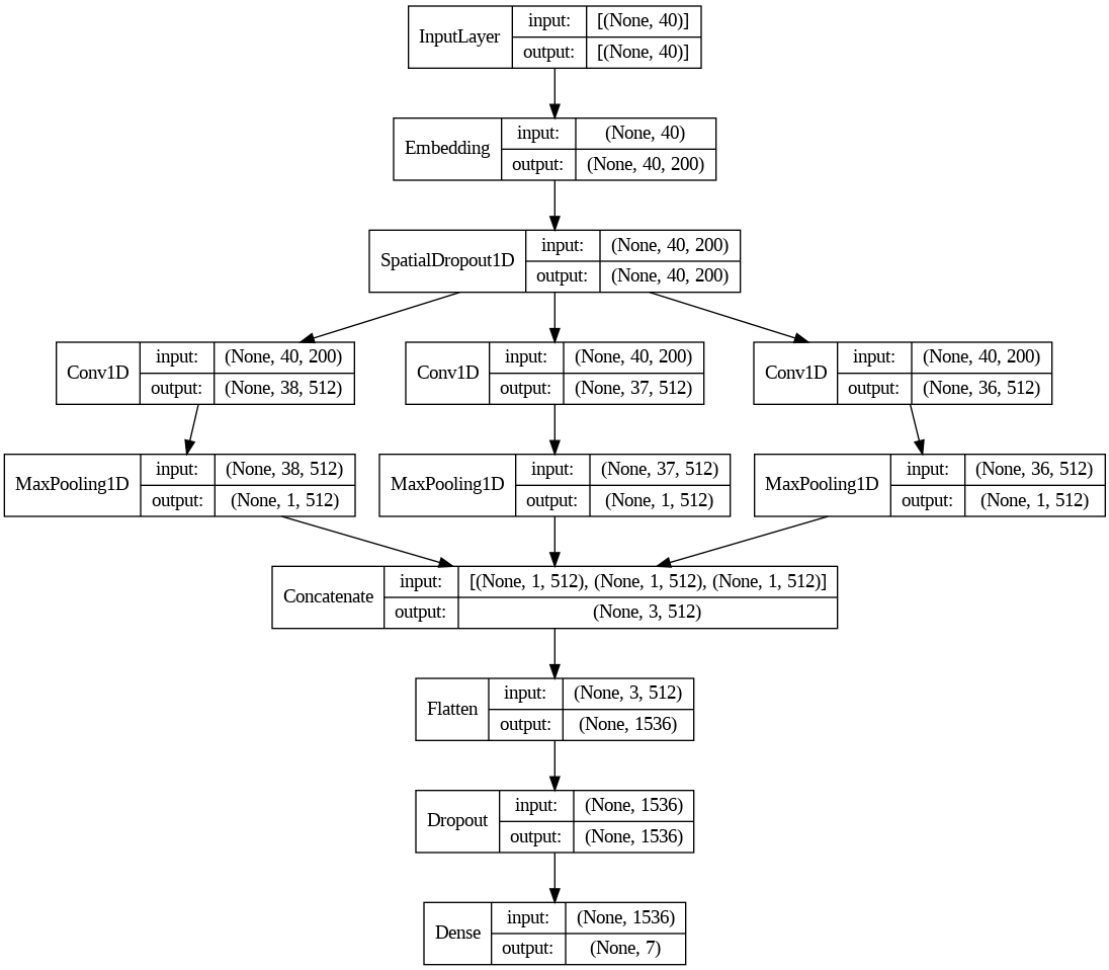


Abbildung A.2: Architektur des CNN-Modells mit einer Spatial-Dropout-Schicht

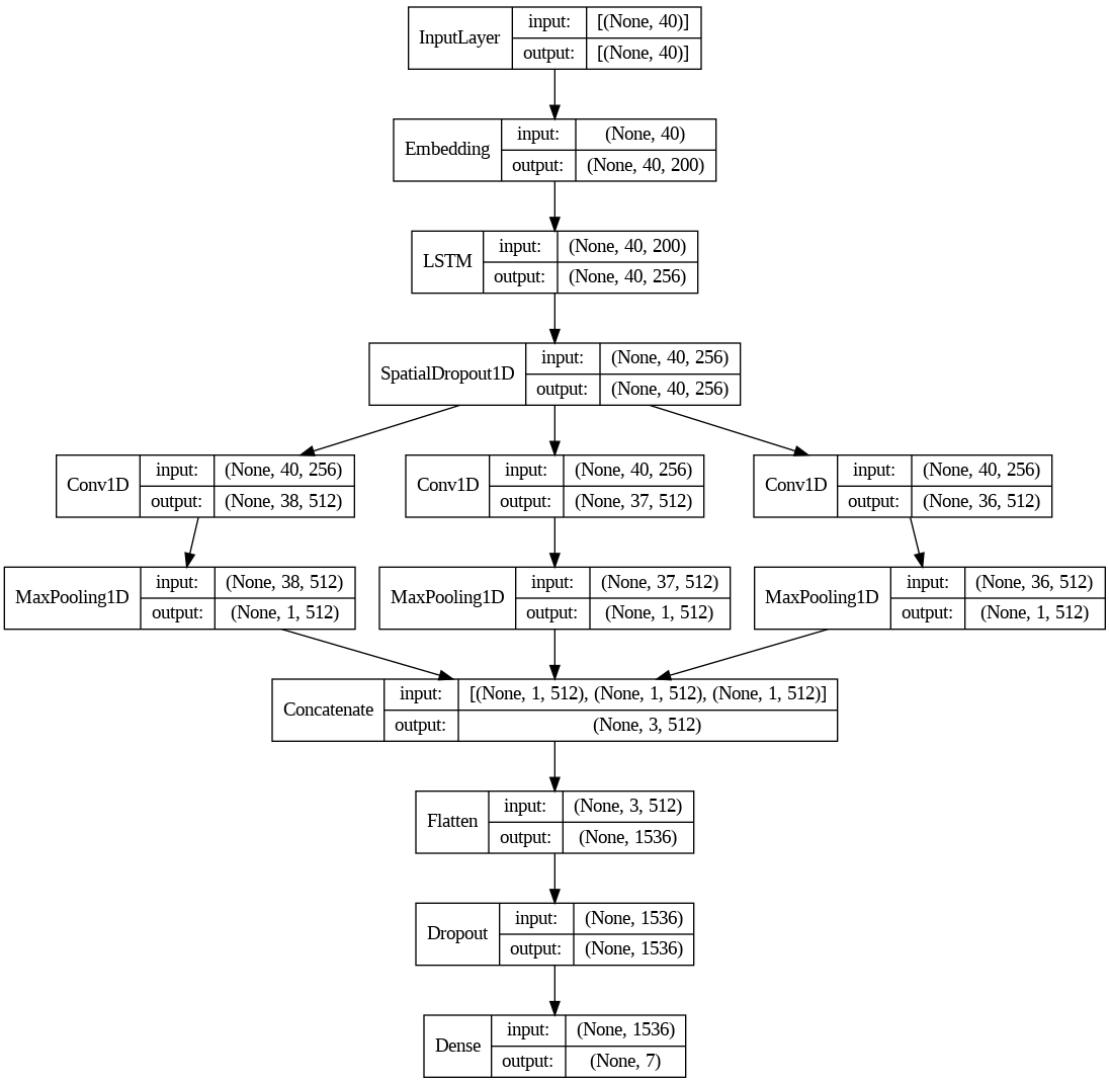


Abbildung A.3: Architektur des LSTM-CNN-Modells mit einer Spatial-Dropout-Schicht

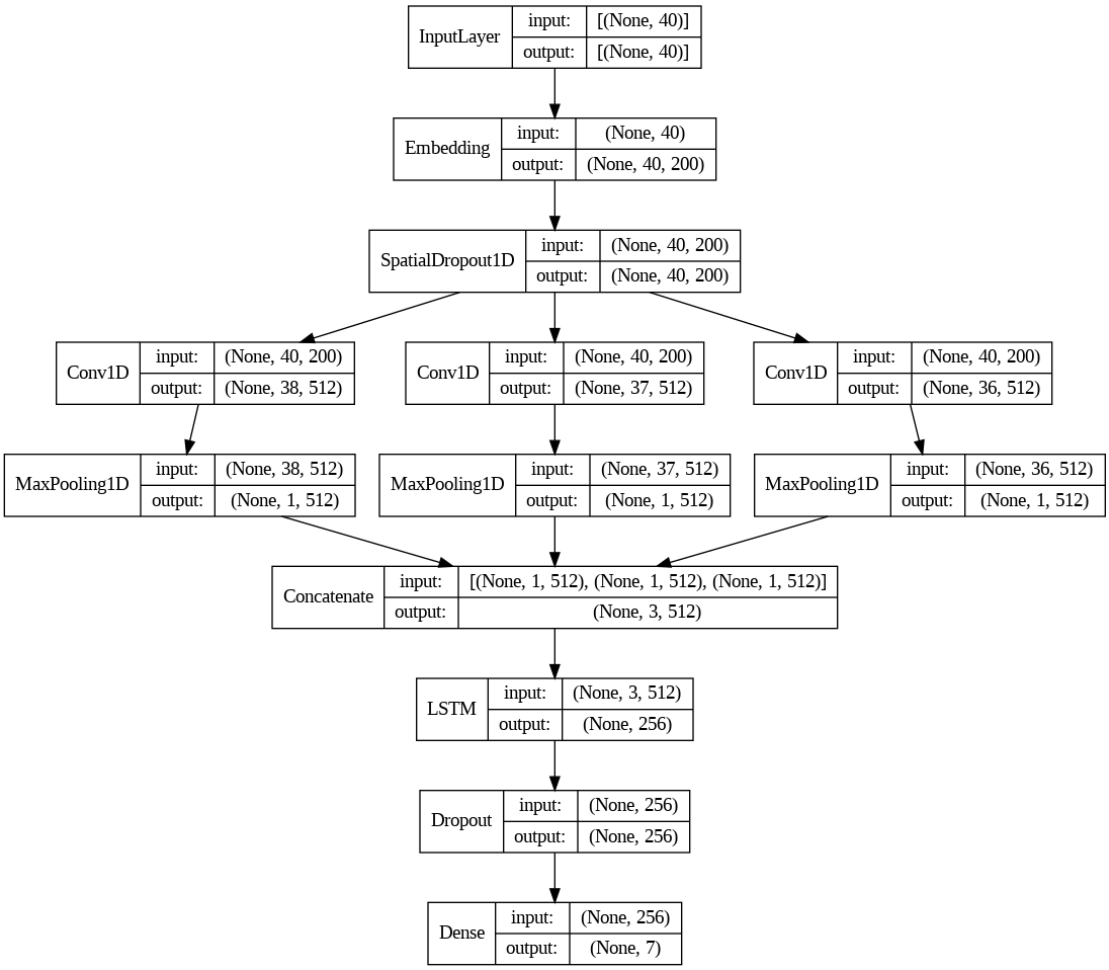


Abbildung A.4: Architektur des CNN-LSTM-Modells mit einer Spatial-Dropout-Schicht

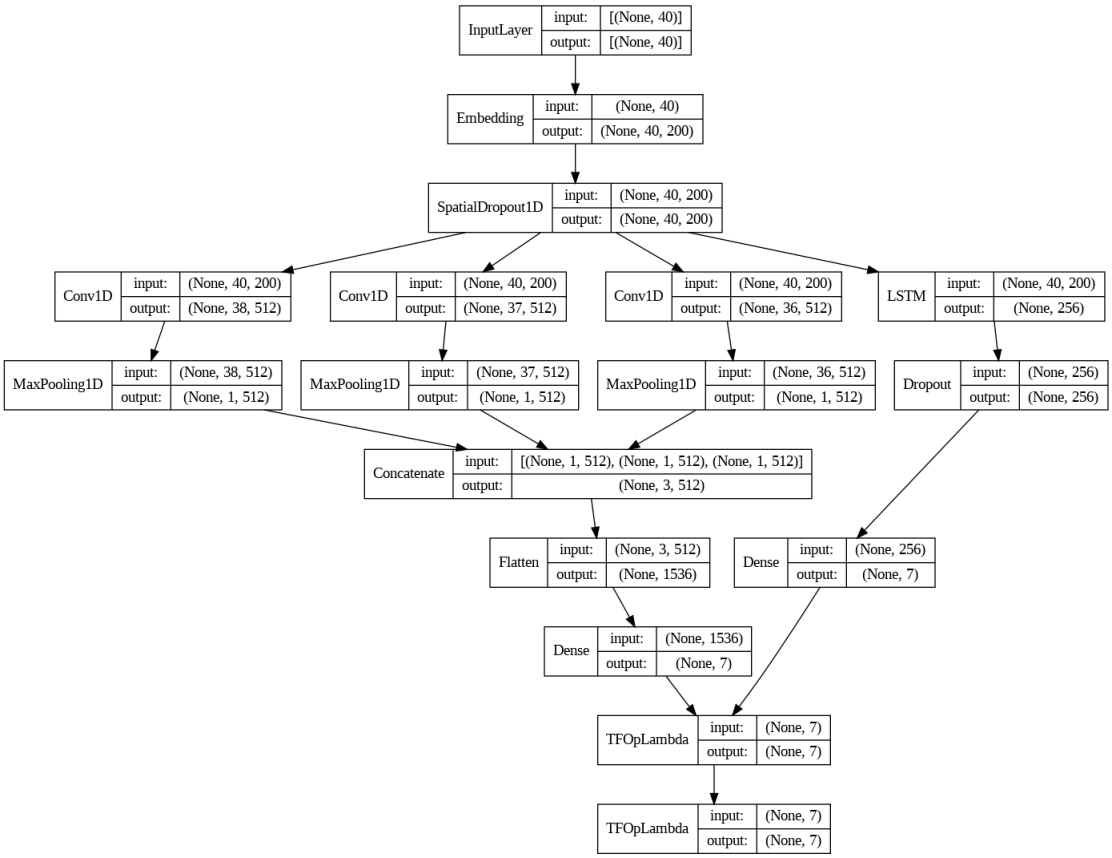


Abbildung A.5: Architektur des Ensembles mit einer Spatial-Dropout-Schicht

A.3 Anzahl der Parameter der Modellen

Die Parameter der Worteinbettungsschichten werden entsprechend der Ausgabegröße der Worteinbettungsschicht angepasst. Die Ausgabegröße der Word2Vec-Worteinbettungsschicht und der GloVe-6B-Worteinbettungsschicht beträgt jeweils 40 * 300, die jeweils 3 859 500 Parameter haben. Zudem ist die Ausgabegröße der GloVe-Tweets-Worteinbettungsschicht 40 * 200, die 2 573 000 Parameter hat.

Modell	Embedding shape	Units	Trainable Params	Non-trainable Params	Total Params
LSTM	40 * 200	128	169 351	2 573 000	2 742 351
		256	469 767		3 042 767
	40 * 300	128	220 551	3 859 500	4 080 051
		256	572 167		4 431 667
CNN	40 * 200	-	1 241 095	2 573 000	3 814 095
	40 * 300	-	1 855 495	3 859 500	5 714 995
LSTM-CNN	40 * 200	128	967 175	2 573 000	3 540 175
		256	2 053 127		4 626 127
	40 * 300	128	1 018 375	3 859 500	4 877 875
		256	2 155 527		6 015 027
CNN-LSTM	40 * 200	128	1 559 431	2 573 000	4 132 431
		256	2 019 591		4 592 591
	40 * 300	128	2 173 831	3 859 500	6 033 331
		256	2 633 991		6 493 491
Ensemble	40 * 200	128	1 410 446	2 573 000	3 983 446
		256	1 710 862		4 283 862
	40 * 300	128	2 076 046	3 859 500	5 935 546
		256	2 427 662		6 287 162

Tabelle A.1: Resultate der Experimente der CNN-LSTM-Modellvarianten

A.4 Erklärungen der Ergebnisse der anderen Modelle mit SP-LIME

Im Abschnitt 6.2 wurden nur die Klassifizierungen des LSTM-CNN-Modells mit SP-LIME erläutert. Im Folgenden sind die Erklärungen für die Klassifizierungen der anderen Modelle.

Abbildung A.6: Erklärungen der Ergebnisse des CNN-LSTM-Modells

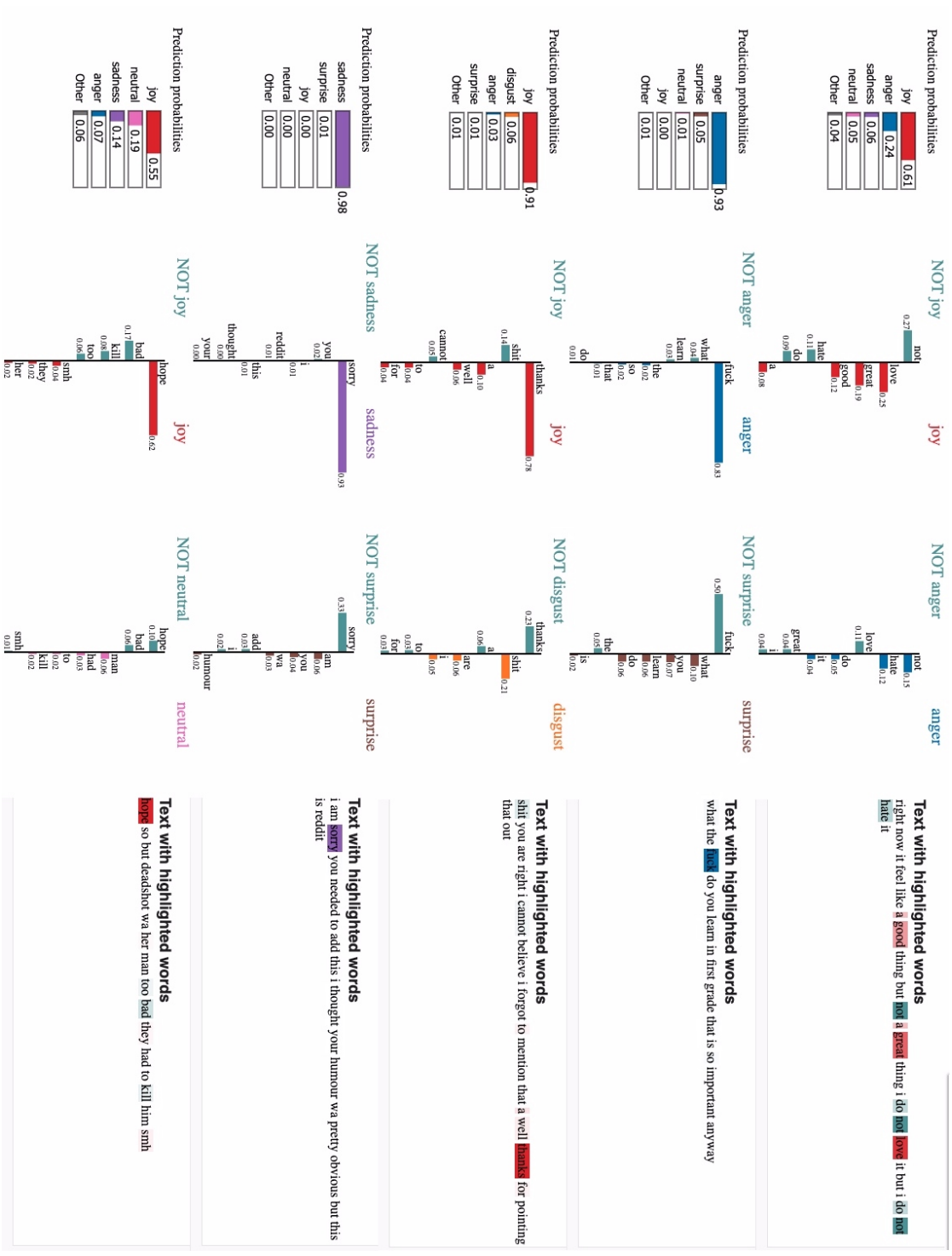


Abbildung A.7: Erklärungen der Ergebnisse des CNN-Modells

Abbildung A.8: Erklärungen der Ergebnisse des LSTM-Modells

Abbildung A.9: Erklärungen der Ergebnisse des Ensemble-Modells

Für jedes Modell wurden 5 Texte aus dem Test-Datensatz ausgewählt und mit SP-LIME erläutert. Die tatsächlichen und vorhergesagten Labels dieser ausgewählten Texte sind in der Tabelle A.2 dargestellt.

Modell	Text Nr.	Label	Predicted Label
CNN-LSTM	1	Sadness	Sadness
	2	Joy	Joy
	3	Neutral	Neutral
	4	Neutral	Joy
	5	Disgust	Joy
CNN	1	Anger	Joy
	2	Anger	Anger
	3	Joy	Joy
	4	Anger	Sadness
	5	Joy	Joy
LSTM	1	Joy	Joy
	2	Neutral	Surprise
	3	Surprise	Joy
	4	Sadness	Fear
	5	Anger	Anger
Ensemble	1	Surprise	Anger
	2	Surprise	Sadness
	3	Joy	Joy
	4	Neutral	Neutral
	5	Anger	Disgust

Tabelle A.2: Tatsächliche und vorhergesagte Labels der ausgewählten Texte

A.5 Konfusionsmatrix der anderen Modelle

Im Abschnitt 6.1.2 wurden die drei besten Modelle mit Konfusionsmatrixen verglichen. Im Folgenden werden das LSTM-Modell und das Ensemble-Modell mit Konfusionsmatrixen analysiert.

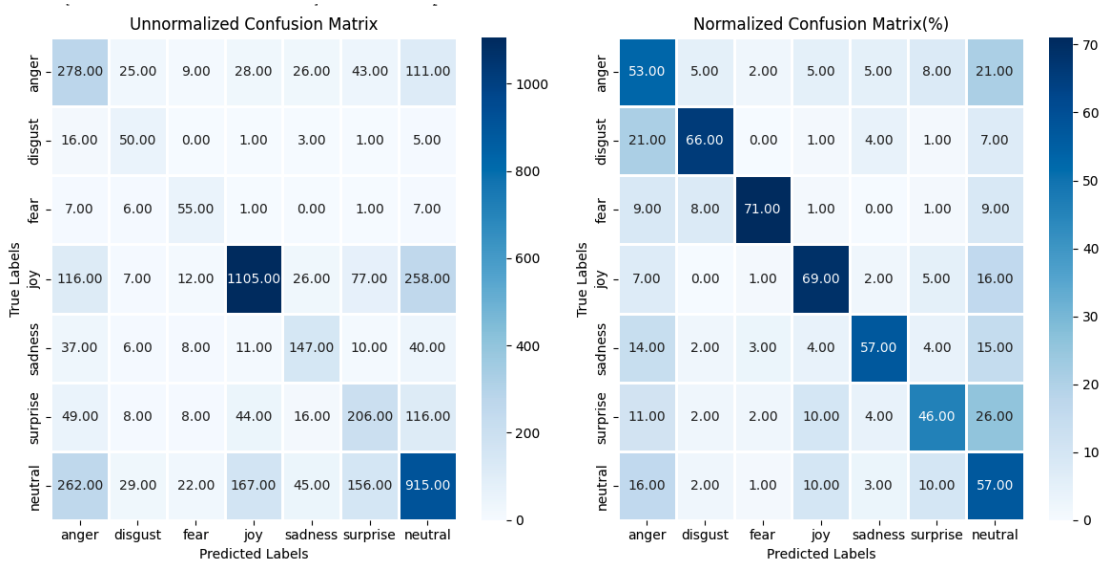


Abbildung A.10: Konfusionsmatrix für das LSTM-Modell

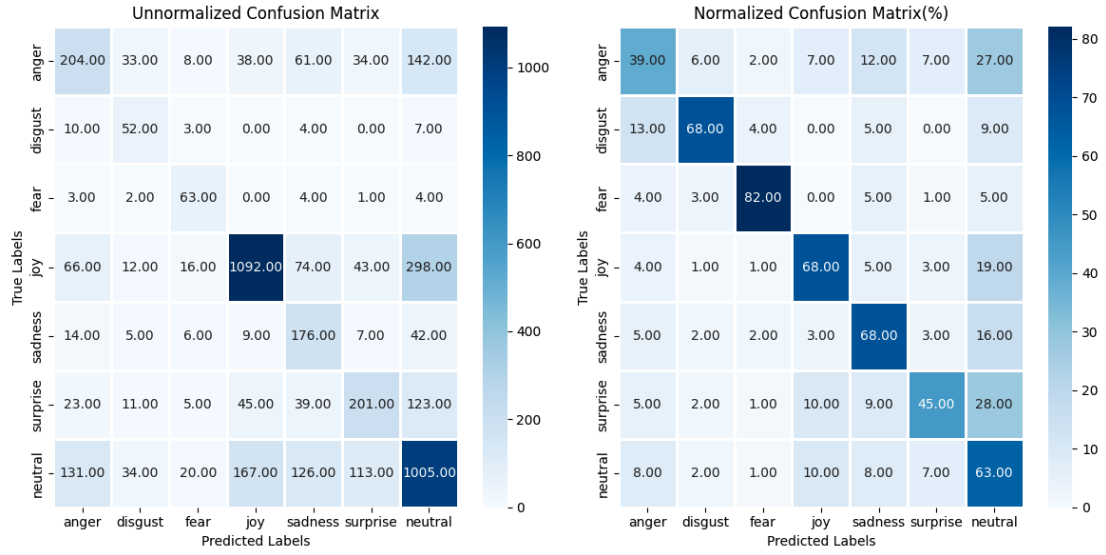


Abbildung A.11: Konfusionsmatrix für das Ensemble-Modell

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original