Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering
InIT Institut für angewandte
Informationstechnologie

Bachelorarbeit Informatik
Frühling 2017

Age und Gender Detection mit Word Embeddings und Deep Learning

Autoren	Florin Hardegger Don Anson Kodiyan
Hauptbetreuer	Dr. Mark Cieliebak Dr. Stephan Neuhaus

Datum	7. Juni 2017

Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

Erklärung betreffend das selbständige Verfassen einer Bachelorarbeit an der School of Engineering

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum:

……

Unterschriften:

……...

……

……

Das Original dieses Formulars ist bei der ZHAW-Version aller abgegebenen Bachelorarbeiten zu Beginn der Dokumentation nach dem Titelblatt mit Original-Unterschriften und -Datum (keine Kopie) einzufügen.

Zusammenfassung

Diese Bachelorarbeit beschäftigt sich mit der Fragestellung, wie sich demografische Merkmale wie Alter oder Geschlecht von Twitter-Benutzern anhand ihrer Tweets möglichst korrekt bestimmen lassen (das sog. Author Profiling-Problem). Author Profiling ist ein wichtiges Instrument der Sprachanalyse und kommt heute bereits zur Anwendung, zum Beispiel um im Marketing die geeignete Zielgruppe für eine bestimmte Werbung zu ermitteln.

Diese Forschungsarbeit ermöglicht es, Alter und Geschlecht von Autoren exakter als in einer vorausgegangenen Projektarbeit vorherzusagen. Ausserdem wurde die Problemstellung erweitert, sodass nun auch die Erkennung von Sprachvariation (Dialekte) möglich ist. Für die Entwicklung der Modelle wurden Daten verwendet, die aus einem wissenschaftlichen Wettbewerbs stammten.

Der Vergleich verschiedener neuronaler Netzwerke und die Verwendung ergänzender Datensätze führt zu einer wesentlichen Verbesserung der Performanz in Author Profiling. Dabei wurde der Fokus auf die Verwendung von Convolutional Neural Networks (CNNs) und Reccurent Neural Networks (RNNs) gerichtet.

Mit unserem System erreichen wir eine Genauigkeit (Anteil korrekter Vorhersagen an allen Vorhersagen) von ca. 50 % bei der Vorhersage des Alters; von fast 80 % bei der Vorhersage des Geschlechts; und von fast 80 % bei der Erkennung des Dialekts englischsprachiger Twitter-Benutzer. Das stellt in allen Bereichen eine deutliche Verbesserung gegenüber den besten Ergebnissen von PAN-2016 dar. Besonders schwierig war dabei die Erkennung des Alters, denn der PAN-Datensatz enthielt nur sehr wenige Benutzer im Alter von über 65 Jahren, was das Training und die Leistung der Modelle stark beeinträchtigt hat. Die Lösung für dieses Problem war die Erweiterung der Datenbasis für diesen Datensatz. Dabei wurden weitere Benutzer hinzugenommen, für die im Twitter-Profil ein Geburtsdatum angegeben war, und bei denen das geschätzte Alter gemäss Profilbild zum angegebenen Alter passte.

Die oben genannten Ergebnisse wurden mit einem Modell aus drei CNNs und einem RNN erzielt. Weitere vielversprechende Resultate liefert der Ansatz mit Bidirectional Gated Recurrent Units mit einem Attention-Mechanismus, der seit einiger Zeit in der maschinellen Übersetzung Bestmarken erreicht. Die Resultate dieses Ansatzes liegen leicht über denen, die wir mit CNNs und RNNs erzielt haben.

Abstract

This thesis addresses the Author Profiling Problem that aims to predict the demographics of Twitter users solely by their tweets. Author Profiling is a task with growing importance and is for example used to determine target groups to deliver personalized advertisement.

This research work makes it possible to predict the age and gender of an author more accurately than in a previous Projektarbeit. In addition, the problem was expanded, so that the recognition of language variation (dialects) is possible. A dataset for this purpose is provided by a scientific competition.

The comparison of different Deep Learning methods and usage of additional data has led to a performance boost in Author Profiling. The focus was set on Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN).

Our system achieves an accuracy of 50% in age detection and almost 80% in the detection of gender and language variety of English Twitter users. This is a major improvement compared to the results of PAN 2016. Age detection has turned out to be the most difficult task, because the dataset contained very few users who are more than 65 years old. This influences our model in a negative way. To solve this issue, additional data was gathered from Twitter. Users who provided their date of birth and a profile picture that roughly ensured the age, were added to our dataset.

The results mentioned above in age and gender detection were achieved with a model consisting of two CNNs and one RNN. Another model with bidirectional gated recurrent units and an attention mechanism achieved the mentioned results in language variety detection. The latter model has achieved remarkable results in machine translation and achieves slightly better results in Author Profiling compared to the first model.

Vorwort

Das Thema Machine Learning hat uns schon in unserer Projektarbeit beschäftigt. Motivation für die Wahl war die Aktualität des Themas sowie die grosse Bandbreite an Forschungsmöglichkeiten. Wie komplex das Thema in der Realität tatsächlich ist, wurde während den Arbeiten an der Projektarbeit deutlich. Schnell war klar, dass wir das Thema deshalb im Rahmen einer Bachelorarbeit vertiefen und uns zugleich der Herausforderung des PAN-Wettbewerbs 2017 stellen wollten.

Ein herzliches Dankeschön gilt unseren Betreuern Dr. Mark Cieliebak, und Dr. Stephan Neuhaus, die uns mit wertvollen Hinweisen unterstützt und begleitet haben. Des Weiteren bedanken wir uns bei Dr. Jörg Keller Paul für die guten Ratschläge zur Struktur und dem Inhalt unserer Arbeit. Ein spezieller Dank geht an Martin Weilenmann, Dirk von Grünigen, Pius von Däniken und Jan Deriu für die interessanten Diskussionen und den Informationsaustausch. Zu guter Letzt danken wir unseren Freundinnen und Familienmitgliedern für die Unterstützung und Geduld.

Inhaltsverzeichnis

1 Einleitung

Soziale Medien haben sich zu einer wichtigen Plattform für die Kommunikation und den Austausch von Informationen entwickelt. Daten können in Sekundenschnelle über den gesamten Globus gesendet werden. Durch die ständige Verfügbarkeit dieser digitalen Kanäle werden auch von den Benutzern deutlich mehr persönliche Daten mit der Öffentlichkeit geteilt. Die stetig wachsenden Datenmengen ermöglichen mehr und immer bessere Analysemöglichkeiten im Bereich Big Data und Machine Learning.

Die verwendete Sprache variiert je nach verwendeter Plattform und gewählten Kommunikationsmittel. In den sozialen Medien ist die Ausdrucksweise aufgrund der Verwendung von Emoticons, Hashtags und limitierter Textgrösse viel individueller als in Briefen oder E-Mails. Entsprechend hat dies im Bereich des Author Profiling die Frage aufgeworfen, ob es möglich ist anhand geschriebener Texte, Schlüsse auf demografische Merkmale der jeweiligen Autoren zu ziehen. Solche Erkenntnisse würden ein immenses Potenzial für zahlreiche Anwendungen beinhalten. So könnte es beispielsweise im Marketingbereich möglich sein, potenzielle Kunden anhand von linguistischer Profile zu ermitteln. Zusätzlich könnten die sozialen Netzwerke anhand von spezifischen Merkmalen Fake-Profile erkennen.

Ziel dieser Arbeit ist, ein Modell zu entwickeln, das die erzielten Author Profiling-Resultate des PAN-16-Wettbewerbs übertrifft. Zusätzlich wollen wir am PAN-17-Wettbewerb in Author Profiling teilnehmen. Für die Teilnahme am PAN-Wettbewerb muss ein Modell entwickelt werden, das für verschiedene Sprachen und Aufgaben gute Resultate liefert. Dafür bauen wir auf den in unserer Projektarbeit gewonnenen Erkenntnissen auf [1].

Für die Zielerreichung, haben wir uns für folgende Experimente entschieden:

- Einfluss von Emotionen und Part-of-Speech (POS) Informationen eines Textes auf die Erkennung von Alter und Geschlecht
- Einfluss State of the Art Techniken des Neuronal Machine Translation (NMT) Forschungsgebiets auf die Author Profiling-Aufgabe
- Bessere Erkennung von Alter durch die Verwendung zusätzlicher Daten
- Bessere Klassifizierung durch die Gruppierung von Tweets eines Autoren
- Leistung der entwickelten Modelle auf der Erkennung von Sprachvariation und die Teilnahme am PAN-Wettbewerb 2017

Diese Arbeit ist wie folgt strukturiert: Kapitel 2 fasst die Ergebnisse ähnlicher Arbeiten zusammen. In Kapitel 3 wird die für diese Arbeit grundlegende Theorie erläutert. In Kapitel 4 wird die Datenbasis und in Kapitel 5 unsere Vorgehensweise erläutert. Der Aufbau und die Resultate aus unseren Experimenten werden in Kapitel 6 beschrieben. Der Abschluss dieser Arbeit bilden die Schlussfolgerung in Kapitel 7 und ein Ausblick über mögliche weiterführende Experimente in Kapitel 8.

2 Stand der Technik

In diesem Kapitel wird der aktuelle Stand der Technik beschrieben. Neuronale Netzwerke (NN) haben in den vergangen Jahren erstaunliche Resultate in der Sprachanalyse erzielt. NNs werden zum Beispiel bereits in Bereichen der Sentimentanalyse [2] und der syntaktischen Analyse [3] intensiv eingesetzt. Probleme wie die Sentimentanalyse und maschinelle Übersetzung erhalten viel Aufmerksamkeit und sind bereits gut erforscht.

Ein allgemeines Problem, das sich auf verschiedene Probleme wie beispielsweise die Sentimentanalyse anwenden lässt, ist die Dokumentenklassifizierung. Dieses Problem wird von Yang et al. [4] angegangen und ihr Netzwerk hat die bisherigen Methoden der Dokumentenklassifizierung deutlich übertroffen. Erreicht wird dies mit einem Hierarchical Network, das mit unterschiedlichen Ebenen arbeitet. Jede Ebene vereinfacht die Informationen und gibt sie der nächsten Ebene weiter. Die einzelnen Ebenen werden mit Recurrent Neural Networks (RNN) und einem Attention-Mechanismus implementiert. Der Attention-Mechanismus wurde von Bahdanau et al. [5] für das Problem der maschinellen Übersetzung entwickelt. Die Arbeit von Bahdanau et al. hatte einen immensen Einfluss, nicht nur auf die maschinelle Übersetzung, sondern auch auf andere Gebiete wie automatisierte Textzusammenfassung [6].

In der Sentimentanalyse hat die Arbeit von Deriu et al. [2] überragende Resultate in der Klassifizierung italienischer Tweets erreicht. Dabei werden word2vec-Repräsentationen eingesetzt, um die einzelnen Wörter abzubilden. Ihr Netzwerk besteht aus einem Convolutional Neural Network (CNN) und benutzt eine Distant Supervised Phase, die das Netzwerk mit einem schwach gelabelten Datensatz trainiert.

Die Forschung in der Anwendung von NNs auf das Author Profiling-Problem ist im Gegensatz zum Sentimentanalyseproblem oder der maschinellen Übersetzung noch sehr jung. In der Geschlechtsklassifizierung wurde von Argamon et al. [7] mit Linear Classifiers gute Resultate erzielt. Ihr System erreicht auf dem British National Corpus eine Genauigkeit von rund 80 %. Die aktuellsten Arbeiten im Bereich des Author Profiling-Problems nutzen bereits NNs. Die Arbeiten von Bartle et al. [8] und Sboev et al. [9] wenden verschiedene Arten von Deep Neural Networks an. Beide Arbeiten haben gezeigt, dass deren Resultate mit dem Stand der Forschung vergleichbar sind. Bartle et al. verwendet eine Kombination aus RNNs und CNNs und erreicht eine Genauigkeit von 86 % in der Geschlechtsklassifizierung auf englischen Blog-Texten. Sboev et al. verwendet ebenfalls CNNs in Kombination mit RNNs. Zusätzlich werden syntaktische Eigenschaften und eine Beschreibung der Emotionen eines Wortes als Feature eingesetzt. Das Modell von Sboev et al. erreicht eine Genauigkeit von rund 86 % auf russischen Texten.

Die Resultate des PAN 2016-Wettbewerbs zeigen, dass Busger et al. in der Altersklassifizierung auf Blogs eine Genauigkeit von rund 59 % und Waser et al. auf Tweets eine Genauigkeit von rund 38 % erreicht [10]. Die Arbeit von Busger et al. verwendet eine Kombination verschiedener stilistischen Features und Waser et al. setzt ein CNN ein.

3 Theoretische Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen, die für das Verständnis dieser Arbeit notwendig sind, beschrieben. Zuerst werden Neuronale Netzwerke im Allgemeinen erklärt, danach Convolutional Neural Networks und Recurrent Neural Networks. Am Schluss werden die Grundlagen für das Trainieren und Evaluieren von Neuronalen Netzwerken erläutert.

3.1 Neuronale Netzwerke

Neuronale Netzwerke (NN) sind ein Versuch, die Funktionalität des menschlichen Gehirns nachzubilden. Im Folgenden werden die wichtigsten Komponenten eines NNs erklärt.

Neuron. Ein *Neuron* ist eine mathematische Funktion und die Einheit eines NNs. Neuronen empfangen n Eingangswerte $\mathbf{x} = (x_0, x_1, \ldots, x_n)$, wobei jedem dieser Eingangswerte x_n ein Gewicht w_n zugewiesen wird. Zusätzlich besitzt ein Neuron eine Ausgabensteuerung x_0, *Bias* genannt. Dieser Bias ermöglicht die Verschiebung der Aktivierungsfunktion, was die Anzahl abbildbarer Funktionen erhöht. Aus den Eingaben x, den Gewichten w und der Aktivierungsfunktion φ berechnet sich die Ausgabe $o(\mathbf{x})$, auch Aktivierung genannt, eines Neurons als:

$$o(\mathbf{x}) = \varphi\left(\sum_{i=0}^{n} w_i x_i\right). \tag{1}$$

Die Aktivierungsfunktion φ sorgt dafür, dass das Resultat o in eine vordefinierte Spanne von Werten gebracht wird. Die in dieser Arbeit verwendeten Aktivierungsfunktionen sind $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, $\mathrm{relu}(x) = \max\{0, x\}$, $sigmoid(x) = \frac{1}{1+e^{-x}}$ und $softmax(x)_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}}$ für $i = 1, \ldots, K$ verschiedene Klassen.

Sigmoid ist eine Spezialfall der logistischen Funktion $f(x) = \frac{L}{1+e^{k(x-x_o)}}$, wobei die Parameter $L = 1$, $k = 1$ und $x_o = 0$ verwendet werden. Die Funktionen werden in der Abbildung 1 dargestellt.

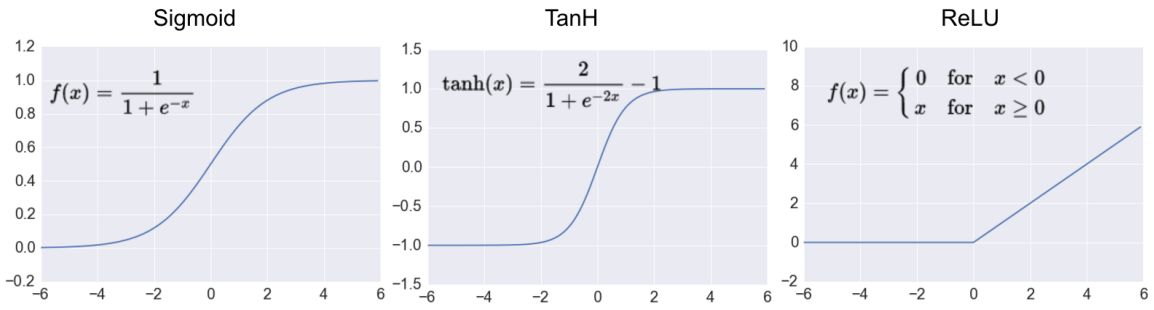

Abbildung 1: **Plots der Aktivierungsfunktionen [11].** Links ist die Sigmoid-Funktion, in der Mitte die Tanh-Funktion und Rechts die Relu-Funktion abgebildet.

Die *Softmax*-Aktivierungsfunktion wird verwendet, um die Werte eines K-Dimensionalen Vektors z in einen neuen K-Dimensionalen Vektor $softmax(\mathbf{z})$ mit Werten im Bereich $(0, 1)$ umzuwandeln, die zusammenaddiert 1 ergeben. Die Funktion lautet:

$$softmax(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{für} \quad j = 1, \ldots, K. \tag{2}$$

Mithilfe der Wahrscheinlichkeitsverteilung über K verschiedene Klassen wird das Resultat der Softmax-Funktion häufig dazu verwendet, Daten in eine Kategorie einzuteilen.

Schicht. Die zuvor beschriebenen Neuronen werden in einem NN in Schichten angeordnet. Eine Schicht setzt sich dabei aus mehreren Neuronen und das NN aus mehreren hintereinander liegenden Schichten zusammen. Dabei wird zwischen drei Typen von Schichten unterschieden: Einer *Eingabeschicht*, einer oder mehreren *verborgene Schichten* und einer *Ausgabeschicht*. Ein Beispiel für ein einfaches NN ist in Abbildung 2 ersichtlich.

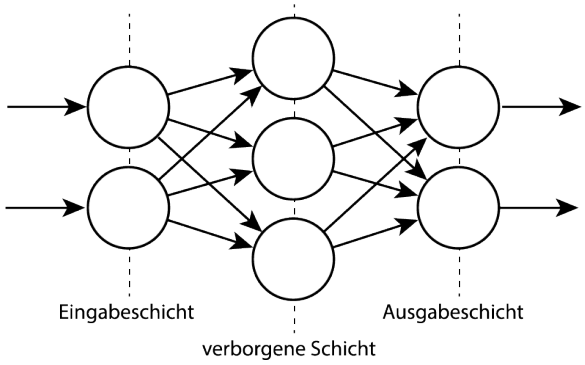

Abbildung 2: **Schematische Darstellung der Schichten in einem einfachen NN.** Links im Bild ist die Eingabeschicht, in der Mitte ist die verborgene Schicht und zum Abschluss wird eine Ausgabeschicht verwendet. Alle Neuronen einer Schicht sind in diesem Beispiel mit allen Neuronen der nächsten Schicht verbunden.

Die Eingabewerte $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ werden über die Eingabeschicht in das NN eingelesen. Der Bias-Wert x_0 beträgt im Normalfall 1 und gilt für alle Neuronen in einer Schicht. Die Neuronen in diesem Netzwerk sind *Fully Connected*, das heisst, dass alle Werte in x an alle Neuronen der nächsten Schicht weitergereicht werden. In der folgenden Schicht wird anhand dieser Werte das Resultat der Aktivierungsfunktion berechnet und an die Ausgabeschicht übermittelt. Die Ausgabe des NNs sind die Resultate der Aktivierung in dieser letzten Schicht.

Jedes Neuron einer Schicht k ist mit allen Neuronen der nächsten Schicht $k + 1$ verbunden. Die Information fliesst ständig vorwärts. Dieser Vorgang wird Vorwärtsprogagierung genannt. Bei einem NN mit mehr als einer verborgenen Schicht spricht man auch von *Deep Neural Network*. Die Anzahl der Schichten und Neuronen in einem NN hängt von der konkreten Problemstellung ab.

Fehlerwert. Der Fehlerwert wird mit einer Fehlerwertfunktion berechnet und verdeutlicht, wie ungenau eine Vorhersage ist. Das Ziel ist es, den Fehlerwert zu minimieren. Um dies

zu erreichen werden Optimizer eingesetzt. Ein Beispiel für einen solchen Optimizer sind Gradient Descent Optimizer, die in Kapitel 3.4 erläutert werden.

Backpropagation. Backpropagation, auch *backward propagation of errors* genannt, wird in Verbindung mit einer Optimierungsmethode wie zum Beispiel Gradient-Descent verwendet, um NN zu trainieren. Der Algorithmus wiederholt dabei die folgenden zwei Phasen:

1. Eingabewerte werden in NNs eingeführt und Schicht für Schicht vorwärts propagiert, bis die Ausgabeschicht erreicht ist. Die Werte der Ausgabeschicht werden mit dem erwarteten Resultat verglichen und daraus die Fehlerwerte der einzelnen Neuronen berechnet.
2. Die Fehlerwerte werden mittels Gradient Descent von der Ausgabeschicht aus durch das NN zurück propagiert, bis jedes Neuron einen Fehlerwert zugewiesen hat. Dieser Fehlerwert entspricht der Beteiligung des Neurons an der ursprünglichen Ausgabe.

Die berechneten Fehlerwerte werden verwendet, um den Gradienten der Verlustfunktion in Bezug auf die Gewichte im NN zu berechnen. In der zweiten Phase wird dieser Gradient dem Optimierungsverfahren zugeführt, das wiederum die Gewichte im NN aktualisiert, um die Verlustfunktion zu minimieren.

Dieser Prozess ermöglicht das Training eines NN, sodass die Neuronen lernen, die unterschiedlichen Eigenschaften der Eingabewerte zu erkennen. Nach dem Training ist das NN in der Lage, in weiteren Daten diejenigen Muster zu erkennen, die es während des Trainings gelernt hat.

Bei der Berechnung der einzelnen Gewichte bestimmt die *Lernrate*, wie stark sich ein einzelner Durchlauf der Backpropagation auf die Gewichte der Neuronen auswirken darf. Bei falscher Wahl der Lernrate könnte das Optimum übersprungen werden oder der Wert der Fehlerfunktion divergieren.

3.2 Convolutional Neural Network

Convolutional Neural Networks (CNN) sind eine Spezialform der im letzten Abschnitt besprochenen NNs. Schichten sind in CNNs nicht mehr Fully Connected, sondern erkennen Muster mittels Filtern über lokale Konnektivität. Nachstehend wird genauer auf die in CNN vorhandenen Komponenten eingegangen.

Filter. CNNs setzen Filter, auch *Kernel* genannt, ein um Muster in den Eingabedaten zu erkennen. Bei Filtern handelt es sich um eine $n \times m$-Matrix. Diese Matrix wird zusammen mit einem Ausschnitt der Eingabedaten verwendet um wichtige Merkmale zu finden. In der Abbildung 3 wird dieser Vorgang grafisch dargestellt.

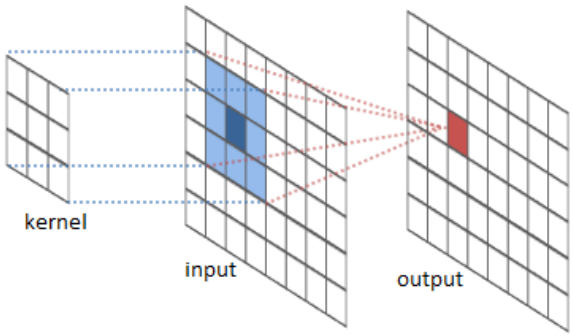

Abbildung 3: **Grafische Darstellung eines Filters [12].** Der Filter (oder Kernel) wird zusammen mit einem Ausschnitt des Inputs verwendet, um den Output zu berechnen.

Diese Filtermatrix wird mit einem *Stride* genannten Bewegungsmuster über die Eingabedaten bewegt. Aus den jeweiligen Ausschnitten wird das Resultat mit der aktuellen Filtermatrix berechnet. Damit sind CNNs beispielsweise in der Lage Buchstaben in einem Bild zu erkennen.

Max Pooling-Schicht. In der Max Pooling-Schicht, eine Art lineares *Downsampling*, werden überflüssige Informationen verworfen. Mit einem Fenster werden die Informationen aus den Feature Maps der vorhergehenden Schicht in Rechtecke unterteilt. Für jedes dieser Rechtecke wird anschliessend das Maximum berechnet und in die gepoolte Repräsentation übernommen. Die Grösse der gepoolten Repräsentation wird durch die Dimensionen des Fensters und den Stride definiert.

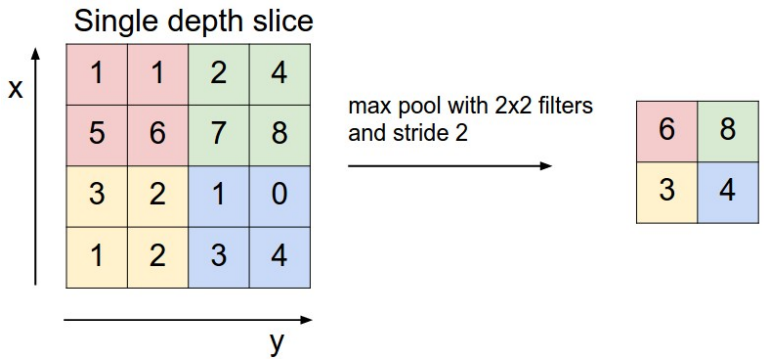

Abbildung 4: **Funktionsweise von Max Pooling [13].** In diesem Max Pooling-Beispiel beträgt die Filtergrösse und der Stride 2. Jedes der 2×2 grossen Rechtecke wird in einer anderen Farbe markiert. Aus jedem Rechteck wird der höchste Wert in die gepoolte Repräsentation übernommen.

Ein Beispiel für Max Pooling ist in Abbildung 4 ersichtlich. Die Werte der $x \times y$ Eingabematrix werden in Rechtecke der Grösse 2×2 aufgeteilt. Für jedes Rechteck wird der höchste Wert in die gepoolte Repräsentation übernommen.

Die gepoolten Resultate einer Feature Map werden als Eingabewerte an die nächste Schicht übergeben. Die Max Pooling-Schicht wird verwendet um die Anzahl der Parameter und Berechnungen in einem CNN zu reduzieren. Zusätzlich wird durch die Verwendung von Max Pooling *Overfitting* kontrolliert.

Convolutional- und Max Pooling-Schicht. CNNs bestehen meistens aus einer beliebigen Anzahl hintereinander gereihten Convolutional- und Max Pooling-Schichten. Ein Beispiel dafür findet sich in Abbildung 5.

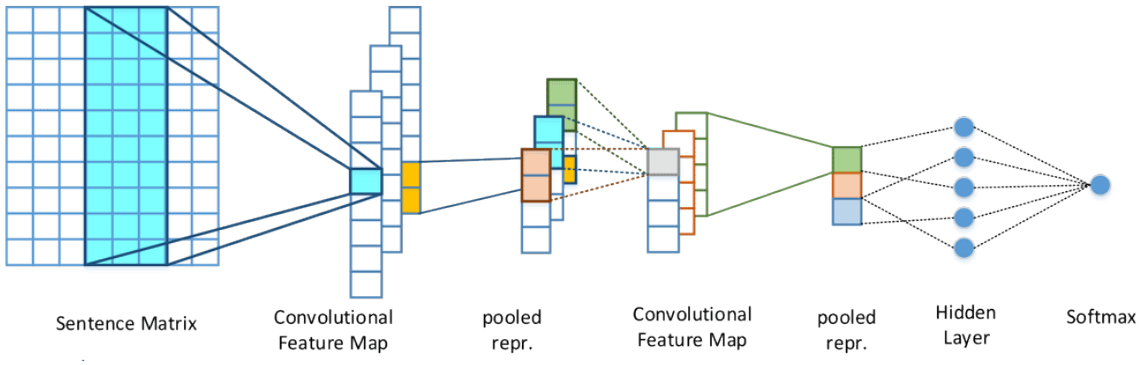

Abbildung 5: **Darstellung eines CNNs mit mehreren Convolutional- und Max Pooling-Schichten [2].** Dieses CNN ist aus einer Kombination von zwei Convolutional- und Max Pooling-Schichten aufgebaut.

3.3 Recurrent Neural Network

Beim Recurrent Neural Network (RNN) handelt es sich um eine Erweiterung des traditionellen NN. Beim traditionellen NN wird davon ausgegangen, dass alle Eingaben unabhängig voneinander sind. Die Eingaben von RNNs sind Sequenzen $\mathbf{x} = (x_0, x_1, \ldots, x_t)$, wobei t für *Time* steht. Mit Time werden die einzelnen Elemente in einer Sequenz bezeichnet. Um zum Beispiel ein Tweet, der eine Sequenz von Wörtern ist, zu klassifizieren, spielt der Zusammenhang zwischen den Wörter eine wichtige Rolle. Deshalb werden für die Berechnungen eines Wortes alle vorherigen Berechnungen miteinbezogen. Dies wird durch die Verwendung eines Speichers ermöglicht. Abbildung 6 zeigt auf, mit welchen Parametern dieser Speicher berechnet wird.

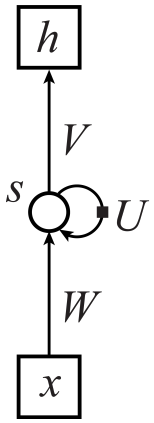

Abbildung 6: **Visualisierte Darstellung eines RNNs.** Bei x handelt es sich um die Eingabesequenz und bei h um die Ausgabesequenz. Für die Berechnung der Ausgabesequenz wird der Speicher s und die Gewichte W, U und V benötigt.

Der Speicher, oft auch *Hidden State* genannt, wird folgendermassen berechnet:

$$s_t = f(Wx_t + Us_{t-1}). \tag{3}$$

Bei f handelt es sich um eine Sigmoid-, Tanh- oder ähnliche Funktion. Die Matrizen W, U und V sind Gewichte, die verwendet werden, um x_t und s_{t-1} zu verändern, bevor sie addiert werden und den neuen Zustand bilden. Die Ausgabe für das Element x_t wird folgendermassen berechnet:

$$o_t = \text{softmax}(Vs_t). \tag{4}$$

Um die Ausgabematrix o_t zu berechnen, wird der Zustand s_t mit der Matrix V multipliziert und auf das Resultat die Softmax-Funktion angewendet. Mit der Ausgabematrix o_t ist man nun in der Lage, das nächste Element vorherzusagen. RNNs sind dadurch in der Lage neue Sequenzen zu generieren und dies macht sie zu einem generativen Modell.

Bei einem Tweet wäre ein RNN in der Lage, das nächste Wort vorherzusagen. Wenn zum Beispiel das erste Wort x_1 'thank' lautet, können RNNs voraussagen, dass das nächste Wort mit grosser Wahrscheinlichkeit x_2 'you' lautet. Wenn das erste Wort x_1 'help' lautet, ist es wahrscheinlicher, dass das nächste Wort x_2 'me' sein wird.

Backpropagation Through Time. *Backpropagation Trough Time* (BPTT) wird für das Trainieren von herkömmlichen RNNs verwendet.

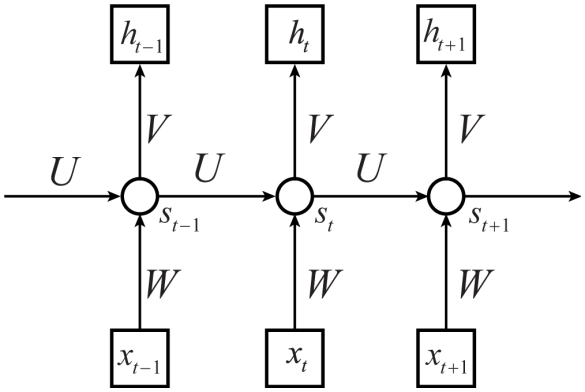

Abbildung 7: **Visualisierte Darstellung eines entfalteten RNNs.** In der entfalteten Darstellung ist die Verwendung der Gewichte W, U und V ersichtlich. x ist die Eingabesequenz, s steht für den Speicher und h ist die Ausgabesequenz.

Bei BPTT muss das RNN zuerst entfalten werden (siehe Abbildung 7), was somit in einem NN resultiert. Danach funktioniert BPTT ähnlich wie die Backpropagation für ein traditionelles NN. Der Unterschied besteht darin, dass die Parameter W, U, V für jede entfaltete Schicht gleich bleiben. Bei der Entfaltung einer langen Eingabesequenz führt dies zu einem langen gewöhnlichen NN. Dies führt zu den im nächsten Abschnitt erklärten Problemen.

Das Vanishing- und Exploding-Gradient-Problem. Das *Vanishing Gradient*-Problem ist ein häufig anzutreffendes Problem bei RNNs. Das Problem liegt darin, dass bei der Backpropagation nach jedem Element der Gradient exponentiell kleiner wird, bis er schliesslich komplett verschwindet. Das *Exploding Gradient*-Problem ist weniger häufig anzutreffen. Die Gradienten verschwinden in diesem Fall nicht, sondern werden sehr gross.

Das Exploding Gradient Problem ist durch Normalisierung einfach zu lösen. Die Vanishing- und Exploding-Gradient-Probleme treten auf, weil einzelne Elemente viele Multiplikationen durchlaufen. Dadurch haben Abhängigkeiten zwischen Elementen über längere Distanzen keinen Einfluss aufeinander. Deshalb sind RNNs schwieriger zu trainieren, weil eine Änderung der Gradienten keine direkte Änderung des Resultates herbeiführt. Um dieses Problem zu lösen, wurden Long Short-Term Memory-Einheiten und Gated Recurrent Units entwickelt, welche in den folgenden Abschnitten genauer erklärt werden.

Long Short-Term Memory. *Long Short-Term Memory*-Einheiten (LSTM) wurden im Jahr 1997 zur Behebung der Probleme der RNNs entwickelt [14]. LSTMs erreichen dies durch die Verwendung eines *Gating*-Mechanismus und sind in der Lage, mit Sequenzen von hunderten Elementen zu trainieren. LSTMs bestehen aus einem Speicher und den drei Gates: Input, Output und Forget. Wie bei einem gewöhnlichen Speicher kann man lesen, schreiben und löschen. Die Gates entscheiden anhand der Eingabedaten, ob gelesen, geschrieben oder gelöscht werden soll. Die Gates werden mit einer logistischen Funktion

implementiert und liefern einen Wert zwischen 0 und 1. In der Abbildung 8 wird der Aufbau einer LSTM-Einheit dargestellt.

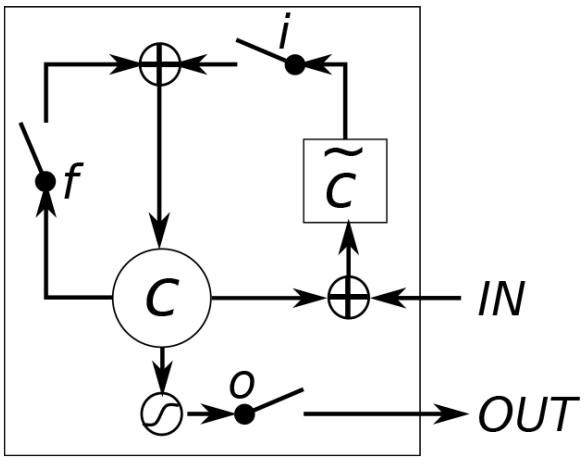

Abbildung 8: **Grafische Darstellung einer LSTM-Einheit [15].** i steht für das Input Gate, f für das Forget Gate und o für das Output Gate. c repräsentiert den Speicher und $\widetilde{c}$ ist eine Zwischenberechnung des neuen Speichers.

Die Ausgabematrix h_t wird in einem LSTM folgendermassen berechnet:

$$h_t = o_t \text{tanh}(c_t). \tag{5}$$

o_t steht für das Output Gate und entscheidet somit, wie viel des Speichers c_t ausgegeben wird. Die drei Gates: Input, Output und Gates i_t, o_t und f_t werden folgendermassen berechnet:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{6}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \tag{7}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f). \tag{8}$$

Bei σ handelt es sich um eine logistische Funktion. W und U sind Gewichte und b ist der Bias. Jedes Gate hat seine eigenen Gewichte und seinen eigenen Bias.

Der Speicher c_t wird wie folgt berechnet:

$$c_t = f_t c_{t-1} + i_t \text{tanh}(W_c x_t + U_c h_{t-1} + b_c). \tag{9}$$

Bei der Berechnung des Speichers wird deutlich, dass das Forget Gate auf den Speicher zugreift. Dies erlaubt dem Forget Gate unwichtige Informationen wieder zu verwerfen. Das Input Gate entscheidet, welche Informationen aus dem neuen Element in den neuen Speicher einfliessen. Um den Speicher c zu berechnen, werden ebenfalls die Parameter W, U und b benötigt.

LSTMs sind sehr gut dafür geeignet etwas zu klassifizieren, weil sie in der Lage sind, sich Informationen über mehrere Elemente hinweg zu merken. Wie zu sehen ist, haben LSTMs jedoch viele Gewichte und benötigen viele Berechnungen. Dies erschwert das Training des LSTMs im Vergleich zum CNN.

Gated Recurrent Unit. *Gated Recurrent Unit*s (GRU) wurden im Jahr 2014 entwickelt und sind LSTMs sehr ähnlich. GRUs sind ebenfalls Erweiterungen eines RNNs und lösen das Vanishing Gradient-Problem mit einem Gating-Mechanismus. GRUs unterscheiden sich von LSTMs dadurch, dass GRUs nur zwei Gates und keinen zusätzlichen Speicher benötigen. In der Abbildung 9 ist der Aufbau einer GRU dargestellt.

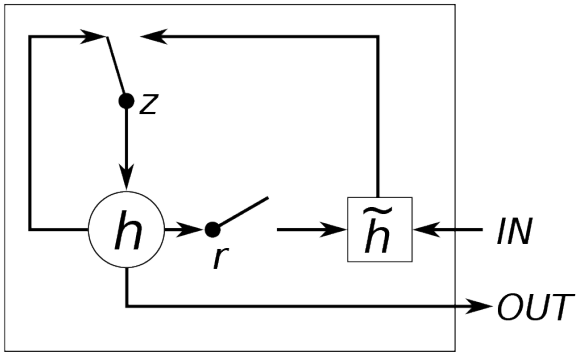

Abbildung 9: **Grafische Darstellung einer GRU [15].** z steht für das Update Gate und r für das Reset Gate. h repräsentiert den Speicher und $\widetilde{h}$ ist eine Zwischenberechnung des neuen Speichers.

GRUs haben ein Update und Reset Gate z_t und r_t, welche folgendermassen berechnet werden:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \tag{10}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r). \tag{11}$$

Der neue Zustand des Speichers h_t wird mit folgender Funktion berechnet:

$$h_t = z_t h_{t-1} + (1 - z_t)\mathsf{tanh}(W_h x_t + U_h(r_t h_{t-1}) + b_c). \tag{12}$$

In der Gleichung 12 ist ersichtlich, dass GRUs im Vergleich zu LSTMs, keinen Output Gate haben. Dies hat zur Folge, dass GRUs immer den kompletten Zustand ausgeben. Dadurch sind GRUs rechnerisch weniger komplex als LSTMs und somit einfacher zu trainieren.

Bidirectional RNN. Gewöhnliche RNNs durchlaufen die Eingabesequenz nur in eine Richtung. RNNs wissen, welche Elemente in einer Sequenz vorkamen, jedoch nicht, welche Elemente noch folgen. Zum Beispiel beziehen sich Wörter in einem Tweet nicht immer auf vorherige Wörter, sondern können sich auf nachstehende Wörter beziehen wie zum Beispiel ein Adjektiv. Damit zukünftige Elemente des RNN dies lernen, werden *Bidirectional Recurrent Neural Network*s (bi-RNN) eingesetzt.

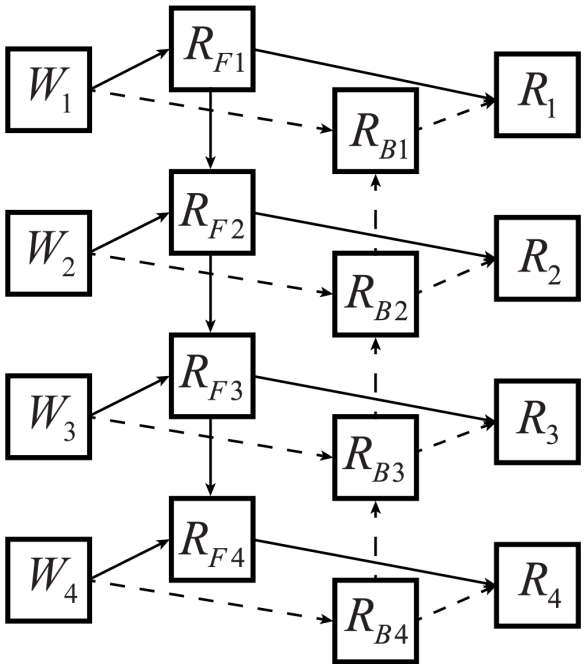

Abbildung 10: **Grafische Darstellung einer bi-RNN-Schicht.** Die bi-RNN-Schicht besteht aus zwei RNN-Schichten: Eine Forward-Schicht R_F und eine Backward-Schicht R_B. Die Eingabe W wird beiden Schichten R_F und R_B übergeben. Durch die Verkettung der Ausgabesequenzen beider Schichten wird die Ausgabe R gebildet.

In der Abbildung 10 ist ersichtlich, dass bi-RNNs zwei gewöhnliche RNN-Einheiten gleichzeitig verwenden. Bei der Ausgabesequenz werden die Ausgaben der beiden RNN-Einheiten miteinander verkettet. Bi-RNNs kennen auf diese Weise die künftigen Elemente.

Bi-RNNs können auch mit LSTM- oder GRU-Einheiten (auch bi-LSTM und bi-GRU genannt) implementiert werden. Um die grosse Menge an Ausgaben zu vereinfachen, gibt es verschiedene Methoden, was in den folgenden Abschnitten beschrieben wird.

Global Max Pooling. Ähnlich wie CNNs können auch RNNs Max Pooling einsetzen. Der Unterschied liegt darin, dass Global Max Pooling nicht mit einem Fenster arbeitet, sondern das grösste Element der Eingabesequenz nimmt. Wenn zum Beispiel Global Max Pooling über die Eingabesequenz $S = [3, 2, 6, 4]$ angewendet wird, wird aus S das Element mit dem grössten Wert zurückgegeben. In diesem Beispiel würde Global Max Pooling 6 zurückgeben.

Diese Methode funktioniert zwar gut, die Informationen werden jedoch stark vereinfacht. Bei der Klassifizierung eines Tweets als Sequenz von Wörtern würde sich das Modell durch Global Max Pooling nur auf die Vektorrepräsentation eines Wortes konzentrieren.

Attention-Mechanismus. Mit dem Attention-Mechanismus wurden im Bereich der Neural Machine Translation (NMT) grosse Fortschritte erzielt. Beim NMT handelt es sich um die maschinelle Übersetzung von Texten mit NNs. Der Attention-Mechanismus ist erfolgreich, weil er zum Beispiel in NMT erkennen kann, welche Wörter beispielsweise für eine Übersetzung wichtig sind.

Die Stärke des Attention-Mechanismus liegt in der Encodierung einer Eingabe. Der Attention-Mechanismus kann in einer Eingabesequenz $S = [w_1, w_2, w_3, w_4]$ erkennen, welche Wörter wichtig sind. Bei der Klassifizierung eines Tweets könnte der Mechanismus zum Beispiel erkennen, dass die Wörter w_2 und w_3 gleich wichtig und die Wörter w_1 und w_4 nicht relevant sind. Die Eingabesequenz S wird deshalb mit der Attention-Matrix $a = [0, 0.5, 0.5, 0]$ multipliziert und anschliessend aufsummiert. In der Abbildung 11 wird der Attention-Mechnismus grafisch dargestellt.

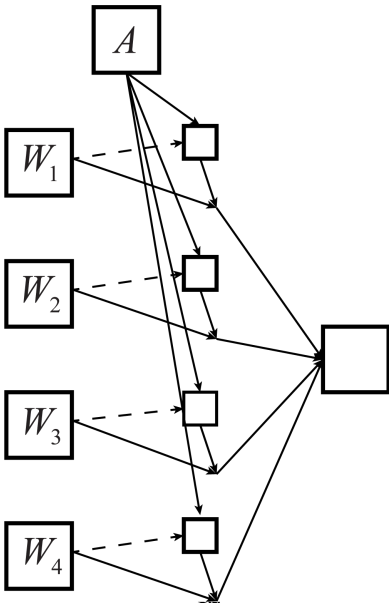

Abbildung 11: **Grafische Darstellung des Attention-Mechanismus.** Mit der Attention Matrix A wird die Gewichtung der encodierten Wörter W_1, W_2, W_3 und W_4 bestimmt. Anschliessend wird die Gewichtung mit den encodierten Wörter multipliziert und aufsummiert.

Um die Gewichtung der einzelnen Elemente zu ermitteln, muss zuerst der verborgene Zustand h_t berechnet werden:

$$h_t = \tanh(Ax + b). \tag{13}$$

Gewicht A und Bias b werden verwendet, um den Zustand für die Eingabesequenz x zu ermitteln. Mit dem Beispiel aus der Abbildung 11 wäre $x = W_1, W_2, W_3, W_4$. Mit dem verborgenen Zustand h_t kann anschliessend die Gewichtung a berechnet werden:

$$a = \text{softmax}(h_t W_u). \tag{14}$$

W_u ist ebenfalls ein Gewicht und a enthält die individuelle Gewichtung (Attention) für die einzelnen Elemente der Eingabesequenz. Um die vereinfachte Eingabe c zu erhalten, müssen a und x multipliziert und anschliessend summiert werden:

$$c = \sum_{t=1}^{n} a_t x_t. \tag{15}$$

Die Ausgabematrix c ist nun eine Vereinfachung der Eingabesequenz x. Wenn die Gleichung 15 genauer betrachtet wird, ist zu erkennen, dass der Attention-Mechanismus ein

gewichteter Durchschnitt ist. Durch diese Gewichtung kann der Attention-Mechanismus die wichtigsten Teile einer Sequenz hervorheben. Eine encodierte Sequenz enthält dadurch relevantere Informationen in einer kompakten Form. Dies macht den Attention-Mechanismus zu einer guten Alternative zu Global Max Pooling.

3.4 Netzwerktraining

Um ein Netzwerk zu trainieren können zwei verschiedene Typen von Daten verwendet werden: *Supervised*- und *Unsupervised*-Daten.

Supervised. Supervised-Datensätze bestehen aus Texten mit den dazugehörigen Annotationen. Diese Annotationen enthalten zum Beispiel Informationen zu Alter und Geschlecht eines Autoren. Während des Trainings werden diese Klassifizierungen benötigt, um den *Fehlerwert* zu berechnen. Alle in dieser Arbeit verwendeten Trainingsdaten gehören in diese Kategorie und werden auch zur Evaluierung unserer Systeme verwendet.

Unsupervised. Unsupervised-Datensätze bestehen aus Texten ohne Annotationen. Zu dieser Kategorie gehören zum Beispiel die Word-Korpus, die zur Erstellung von Word-Embeddings verwendet werden.

Gradient Descent Optimizers. Gradient Descent Optimizer sind iterative Algorithmen und gehören zu den beliebtesten Methoden, um NN zu trainieren. Gradient Descent Optimizer werden eingesetzt, wenn es nicht möglich ist, die Parameter algebraisch zu berechnen. Das Ziel der Optimizer ist es, ein Minimum für eine Funktion zu finden, indem die Parameter angepasst werden. Die Parameter werden nach jeder Iteration angepasst, dazu wird eine Teilmenge der Trainingsdaten benötigt. Diese Teilmenge wird als *Batch* bezeichnet. Ein Durchlauf der gesamten Trainingsdaten wird als *Epoche* bezeichnet.

Eine Funktion kann als eine unebene Landschaft betrachtet werden. Der Optimizer hat die Aufgabe den tiefsten Punkt dieser Landschaft zu ermitteln.

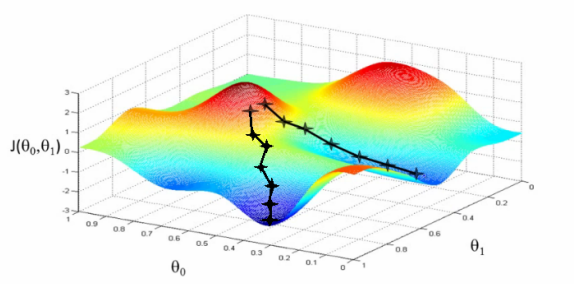

Abbildung 12: **Grafische Darstellung einer Funktion und zwei mögliche Pfade eines Optimizers [16].** Der Optimizer versucht den kleinsten Wert für die Funktion J zu finden indem er die Parameter θ_0 und θ_1 anpasst.

In der Abbildung 12 ist ersichtlich wie eine iterative Suche nach dem Minimum aussehen könnte. In dieser Arbeit werden zwei Gradient Descent Optimizer AdaDelta [17] und Adam [18] verwendet. Beide Algorithmen haben den Vorteil, dass sie wenig bis keine Hyperparameteranpassungen benötigen und sich die Lernraten automatisch in Abhängigkeit der vorherigen Durchläufe anpassen.

Overfitting. Overfitting erfolgt, wenn beim Trainieren eines Netzwerks die Parameter stark an die Trainingsdaten angepasst werden. Ein solches Netzwerk erzielt gute Resultate auf den Trainingsdaten, kann jedoch auf neuen Daten keine guten Resultate erzielen. Dies ist vergleichbar mit einem Schüler, der sich auf eine Prüfung vorbereitet, indem er eine Menge von Additionen auswendig lernt, wie man addiert lernt er jedoch nicht. Wenn jedoch an der Prüfung andere Additionsaufgaben auftauchen, kann der Schüler diese Aufgaben nicht lösen. Um Overfitting zu verhindern, werden *Regularizers* eingesetzt, welche in den nächsten Abschnitten beschrieben werden.

Early Stopping. Early Stopping ist eine Art von *Regularizer* und verwendet Validierungsdaten um die Leistung des Modells zu messen. Die besten Gewichte werden anhand der Validierungsdaten ausgewählt. Validierungsdaten werden mit der Idee verwendet, dass ein Modell auf den Testdaten ähnlich funktioniert wie auf den Validierungsdaten. Mit Early Stopping wird solange trainiert, bis keine besseren Gewichte gefunden werden.

Dropout. Dropout ist ebenfalls eine Art von Regularizer und verwirft zufällig Neuronen in einem NN während dem Training. Wie viele Neuronen verworfen werden, wird durch die Drop-Rate definiert. Mit Dropout wird verhindert, dass sich ein Netzwerk stark an die Trainingsdaten anpasst.

Class Weights. Bei unausgeglichenen Datensätzen fokussieren sich NNs beim Trainieren auf die stark vertretenen Klassen. So kann es passieren, dass ein Netzwerk zu einseitige Vorhersagen trifft. Um das zu verhindern, werden Class Weights eingesetzt. Sie skalieren die Fehlerwertfunktion, damit jede Klasse ausgeglichen trainiert wird.

3.5 Evaluierung

Die Leistung unserer Modelle wurde mit den Evaluierungsmetriken Genauigkeit und F1 Score gemessen. Anhand des erzielten *F1 Scores* werden die Gewichte der Modelle als *Model Checkpoint* abgespeichert und anhand der erzielten *Genauigkeit* werden die effektive Leistungen der Modelle verglichen und bewertet. Die in den Formeln verwendeten Abkürzungen tp, tn, fp, fn stehen für *true positives*, *true negatives*, *false positives* und *false negatives*.

Genauigkeit. Die Genauigkeit (Englisch Accuracy) berechnet sich aus dem Verhältnis korrekt klassifizierter Datensätze $tp + tn$ zu allen Datensätzen $tp + tn + fp + fn$:

$$acurracy = \frac{tp + tn}{tp + tn + fp + fn}. \tag{16}$$

Genauigkeit ist eine einfache Möglichkeit die Performanz von Modellen mit einer oder mehreren Klassen gleicher Grösse zu bestimmen. Sind die Klassen jedoch ungleich verteilt, lassen sich gute Resultate erzielen, indem immer die am stärksten vertretene Klasse vorhergesagt wird. Diese Besonderheit muss man bei der Betrachtung der Resultate beachten.

F1 Score. Der F1 Score unterstützt die Klassifizierung in unregelmässig verteilten Daten und ermöglicht eine aussagekräftige Evaluierung. Der F1 Score berechnet sich aus den beiden Metriken *Präzision* (Englisch Precision) und *Ausbeute* (Englisch Recall). Dabei wird die Präzision aus dem Verhältnis richtig klassifizierten Datensätzen tp zu allen klassifizierten Datensätzen $tp + fp$ einer Klasse berechnet:

$$precision = \frac{tp}{tp + fp}. \tag{17}$$

Die Ausbeute ist das Verhältnis zwischen richtig klassifizierten tp und allen vorhandenen Datensätzen $tp + fn$ in dieser Klasse:

$$recall = \frac{tp}{tp + fn}. \tag{18}$$

Der F1 Score, der auch gewichteter Durchschnitt genannt wird, ist das harmonische Mittel von Präzision und Ausbeute:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}. \tag{19}$$

Ein Modell lässt sich sowohl mit Präzision und Ausbeute, als auch mit dem F1 Score evaluieren. Der F1 Score bietet den Vorteil, dass Modelle anhand von nur einem Wert bewertet werden können.

Um ein Modell über mehrere Klassen zu bewerten, berechnet man die Summe der einzelnen F1 Scores über die verschiedene Klassen k_i und dividiert diese Zahl durch die Anzahl Klassen n:

$$F_{1_{k_0,k_1,\ldots,k_n}} = \frac{\sum_{i=0}^{n} F_{1_{k_i}}}{n}. \tag{20}$$

Der F1 Score über mehrere Klassen wird auch *Macro-averaged* F1 Score genannt. In dieser Arbeit wird immer der Macro-averaged F1 Score verwendet.

Cross Validation. Cross Validation ist eine Möglichkeit zur Validierung von Modellen. Im Training von Machine Learning-Modellen wird mit dieser Technik überprüft wie ein Modell auf unbekannte Daten reagiert. Dazu werden die Daten während des Trainings in zwei Teile aufgeteilt: Training und Validierung. Das Ziel von Cross Validation ist es, Probleme wie *Overfitting* zu verhindern und aufzuzeigen wie das Model auf unbekannten Daten abschneidet.

Bei Cross Validation wird zwischen zwei verschiedenen Typen unterschieden: *Exhaustive Cross Validation* und *non-exhaustive Cross Validation*.

Exhaustive Cross Validation. Bei *exhaustive Cross Validation* Methoden wird darauf geachtet, dass die Daten in alle mögliche Kombinationen von Training und Testdaten aufgeteilt werden. Es kann vorkommen, dass alle Daten einer Iteration aus der gleichen Klasse stammen.

Non-exhaustive Cross Validation. Bei diesem Cross Validation-Typ werden die Testdaten nicht in alle Kombinationen aufgeteilt, sondern darauf geachtet, dass die Klassen sinnvoll auf die einzelnen Iterationen verteilt sind. Eine Methode des non-exhaustive Cross Validation ist *k-Fold Cross Validation* und wird im nächsten Abschnitt erklärt.

K-Fold Cross Validation. Um Schwankungen zu vermeiden, werden in der k-Fold Cross Validation-Methode mehrere Iterationen Cross Validation durchgeführt und dabei jedes Mal andere Daten zur Validierung verwendet. Wie die Daten aufgeteilt werden könnten, wird in Abbildung 13 gezeigt.

Bei dieser Methode ist es wichtig, dass dieselben Daten in nur einem Fold (Iteration) als Testdaten verwendet werden dürfen. Nach k Iterationen Cross Validation werden die Resultate der einzelnen Iterationen gemittelt.

Ein Vorteil von k-Fold Cross Validation ist, dass man es anwenden kann, wenn für die konventionelle Validierung (Aufteilung der Daten in 70% Training und 30% Testdaten) zu wenig Daten vorhanden sind. Cross Validation bietet auch in diesen Fällen eine Möglichkeit zur Evaluierung eines Modells.

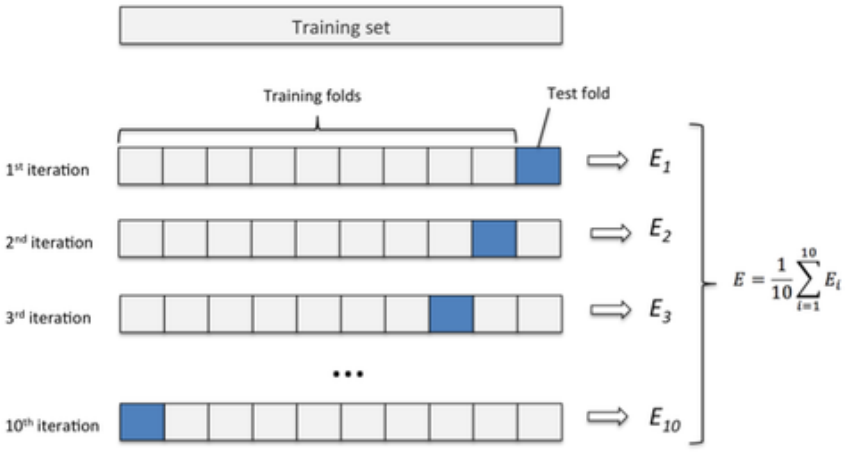

Abbildung 13: **Aufteilung der Daten in den einzelnen Iterationen mit k-Fold Cross Validation [19].** Aufteilung der Daten in k-Fold Cross Validation mit $k = 10$. In jeder Iteration werden andere Daten als Testdaten verwendet. Die Resultate der einzelnen Iterationen E_i werden am Schluss summiert und gemittelt.

Stratified k-Fold Cross Validation. Mit *Stratified k-Fold Cross Validation* achtet man bei der Verteilung der Daten auf die einzelnen Folds zusätzlich darauf, dass die Klassen gleichmässig auf die Folds verteilt sind.

4 Daten

In diesem Kapitel werden die in dieser Arbeit verwendeten Daten genauer beschrieben. Es wird erläutert, woher die Daten stammen, wie sie verwendet werden und wie sie aufgebaut sind. Ausserdem wird erklärt, woher die Word-Embeddings stammen und wie sie generiert werden.

4.1 Trainingsdaten

In dieser Arbeit wurden drei verschiedene Trainingsdatensätze verwendet. Zwei davon wurden im Rahmen des in Kapitel 9.1 beschriebenen PAN Shared Task zur Verfügung gestellt und ein Datensatz wurde von den Autoren dieser Arbeit hinzugefügt.

PAN Author Profiling Datensätze. Für die Teilnehmer der PAN Shared Tasks werden Trainingsdaten zur Verfügung gestellt. Die Trainingsdaten werden jedes Jahr auf die jeweiligen Aufgaben abgestimmt.

PAN-2016 Trainingsdaten. Im Jahr 2016 lag der Fokus des Author Profiling-Tasks auf der Bestimmung von Alter und Geschlecht eines Twitter Autoren basierend auf seinen Tweets. In der Folge bestand dieser Datensatz aus Tweets, die mit Geschlecht und Alter annotiert wurden. Mit der Ausnahme der holländischen Tweets, die als einzige keine Informationen zum Alter der jeweiligen Autoren enthalten. Der Datensatz beinhaltet Tweets von total 1070 Autoren, verteilt auf drei Sprachen: Englisch, Spanisch und Holländisch. Davon wurden in dieser Arbeit nur die Tweets in englischer Sprache verwendet. Die genaue Verteilung der Daten ist in Tabelle 1 ersichtlich.

Sprache	**18-24**	**25-34**	**35-49**	**50-64**	**65-XX**	**Total**
Englisch	28	140	182	80	6	436
Spanisch	16	64	126	38	6	250
Holländisch	keine Information					384
Total	**44**	**204**	**308**	**118**	**12**	**1070**

Tabelle 1: **Verteilung der Profile im PAN-2016-Trainingsdatensatz.** Alle Klassen beinhalten gleich viele weibliche und männliche Profile.

In jeder Alterskategorie sind gleich viele weibliche und männliche Profile vorhanden. Die Anzahl Tweets der einzelnen Autoren variiert jedoch zwischen den Klassen, was in der Tabelle 2 deutlich wird.

Englisch	18-24	25-34	35-49	50-64	65-XX	Total
Weiblich	9261	42'489	45'095	21'883	474	119'202
Männlich	9931	52'852	67'934	23'512	2040	156'269
Total	**19'192**	**95'341**	**113'029**	**45'395**	**2514**	**275'471**

Tabelle 2: **Anzahl Tweets in den englischen Daten.** Die Tweets sind nach Alterskategorie und Geschlecht gruppiert.

Die Anzahl Tweets variiert zwischen den Alterskategorien und zwischen den Geschlechtern stark. So sind zum Beispiel in der 65-XX Kategorie fünf mal mehr männliche wie weibliche Tweets enthalten.

Der PAN-2016 Trainingsdatensatz ist der letzte Datensatz des Wettbewerbs, der Informationen zum Alter eines Autoren enthält. Aus diesem Grund wird dieser Datensatz weiterhin in dieser Arbeit verwendet.

PAN-2017-Trainingsdaten. Im PAN-2017-Wettbewerb wurde ein neuer Shared Task eingeführt. Anstelle von Alter muss nun die Sprachvariationen von Twitter-Benutzern erkannt werden. Der Datensatz enthält dementsprechend Informationen zu Geschlecht und Sprachvariationen in vier verschiedenen Sprachen: Englisch, Spanisch, Portugiesisch und Arabisch. Auch enthält der PAN-2017-Datensatz teilweise Autoren aus dem PAN-2016-Datensatz. Eine detaillierte Verteilung der Daten ist in Tabelle 3 dargestellt.

Sprache	Anzahl Profile	Anzahl Sprachvariationen	Sprachvariationen (600 Autoren pro Sprachvariation)
Englisch	3600	6	Australien, Kanada, Grossbritannien, Irland, Neuseeland, USA
Spanisch	4200	7	Argentinien, Chile, Kolumbien, Mexiko, Peru, Spanien, Venezuela
Portugiesisch	1200	2	Brasilien, Portugal
Arabisch	2400	4	Ägypten, Golf, Levante, Westarabisch
Total	**11'400**	**19**	

Tabelle 3: **Verteilung der Profile im PAN-2017-Trainingsdatensatz.** Alle Klassen beinhalten gleich viele weibliche und männliche Profile.

Autoren und Tweets sind in diesem Datensatz gleichmässig verteilt. Für jeden Autoren wurden genau 100 Tweets gesammelt und jede Kategorie besteht aus der gleichen Anzahl weiblicher und männlicher Profile. Für jede der Sprachvariationen liegen ausserdem genau 600 Autoren vor.

Mit diesem Datensatz wurden alle in dieser Arbeit verwendeten Modelle zur Erkennung von Geschlecht und Sprachvariation trainiert und getestet.

Hardiyan-2017-Datensatz. Für diese Arbeit wurden neue Daten gesammelt und mit diesen ein neuer Datensatz erstellt. In diesem Datensatz ist das genaue Alter (Geburtstag) und das Geschlecht der Autoren bekannt. Mit diesen Informationen wurden die Tweets der Autoren in die richtige Kategorie eingeordnet. Es ist möglich, dass ein Autor in mehr als einer Alterskategorie vertreten ist, je nach Alter des Autors zum Verfassungszeitpunkt seiner Tweets.

Mit dieser Methode wurden 5052 Profile gefunden. Die genaue Verteilung ist in Tabelle 4 ersichtlich.

Englisch	**18-24**	**25-34**	**35-49**	**50-64**	**65-XX**	**Total**
Männlich	186	1726	1113	242	42	3309
Weiblich	140	1071	455	70	7	1743
Total	**326**	**2791**	**1568**	**312**	**49**	**5052**

Tabelle 4: **Verteilung der Profile im Hardiyan-2017-Datensatz.** Autoren können in mehreren Alterskategorien vertreten sein, je nach Alter des Autors zum Erfassungszeitpunkt des jeweiligen Tweets.

4.2 Word-Embeddings

In diesem Kapitel werden die in dieser Arbeit verwendeten Word-Embeddings beschrieben. Es werden zwei verschiedene Arten Word-Embeddings verwendet, einerseits *FastText* Word-Embeddings [20] und anderseits *word2vec* Word-Embeddings [21]:

FastText Word-Embeddings. In den beiden Sprachen Portugiesisch und Arabisch wurden vortrainierte FastText Word-Embeddings von Facebook[1] verwendet. Die Embeddings wurden auf Wikipedia[2] Texten trainiert. Die dazugehörigen Hyperparameter sind in der Tabelle 5 aufgelistet.

Name	**Wert**
algorithm	skip-gram
dimensions	$d = 300$
window size	5
minimum count	5

Tabelle 5: **Liste der Hyperparameter zur Erzeugung der FastText Word-Embeddings.**

Word2vec Word-Embeddings. Für die Experimente in den beiden Sprachen Englisch und Spanisch wurden die word2vec Word-Embeddings der Firma SpinningBytes AG[3] verwendet [22]. Die Word-Embeddings der Dimension $d = 52$ wurden mit Hilfe eines 590 Millionen Tweet umfassenden Corpus und die Embeddings der Dimension $d = 200$ mit einem 200 Millionen Tweet umfassenden Corpus erstellt. Die Embeddings wurden mittels word2vec generiert und die in Tabelle 6 aufgelisteten Hyperparameter verwendet.

[1] http://www.facebook.com
[2] http://www.wikipedia.org
[3] http://spinningbytes.com/resources/embeddings/

Name	Wert
algorithm	skip-gram
dimensions	$d = 52$ oder $d = 200$
window size	5
minimum count	15

Tabelle 6: **Liste der Hyperparameter zur Erzeugung der word2vec Word-Embeddings.**

Im Rahmen dieser Arbeit wurden sowohl die Word-Embeddings der Dimension $d = 52$ als auch $d = 200$ verwendet. Bei der Erstellung dieser Word-Embeddings unterscheiden sich nur die Dimension und die Anzahl verwendeter Tweets, ansonsten bleiben die Parameter unverändert.

4.3 Emotion Lexika

In dieser Arbeit wurde ebenfalls untersucht, ob die Emotionalität eines Tweets eine Variable sein könnte, um die Genauigkeit der Bestimmung von Alter und Geschlecht eines Autoren zu steigern. Für dieses Experiment wurde die Version 0.92 des *NRC Word-Emotion and Word-Sentiment Association Lexicon* (EmoLex) [23], [24] verwendet.

EmoLex enthält Annotationen zu acht verschiedenen Emotionen: Angst, Erwartung, Ekel, Wut, Freude, Traurigkeit, Überraschung und Vertrauen. Zusätzlich enthält das Lexikon Annotationen zu den Sentimenten der Wörter. Ein Sentiment beschreibt das Empfinden, welches beim Lesen eines Wortes ausgelöst wird. Es wird zwischen den Sentimenten positiv, negativ und neutral unterschieden.

Im Lexikon sind total 14'182 englische Unigramme (Wörter) enthalten, die manuell annotiert wurden. Die Häufigkeit der einzelnen Emotionen wird in Abbildung 14 gezeigt.

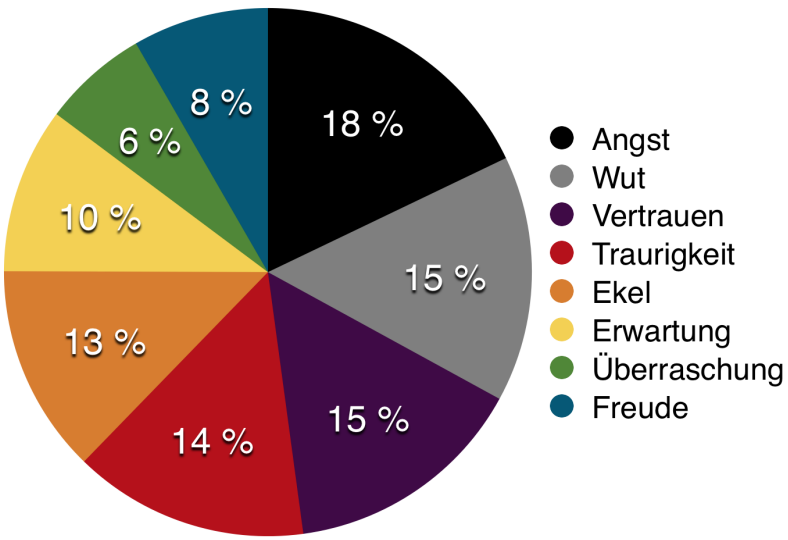

Abbildung 14: **Häufigkeiten der acht Emotionen im EmoLex.**

In 8290 der 14'182 Wörter wurde eine Emotion erkannt und annotiert. In den restlichen 5892 Wörter sind nur Informationen zum Sentiment verfügbar. Die Häufigkeiten in Abbildung 14 beziehen sich dementsprechend auf 8290 Wörter.

5 Methodik

In diesem Kapitel wird in einem ersten Schritt die Forschungsmethodik erklärt. Anschliessend wird die Standardeinrichtung der Experimente beschrieben.

5.1 Vorgehensweise

Für das Ziel dieser Arbeit waren fünf Fragestellungen handlungsleitend. Jede dieser Fragestellungen war jeweils die Basis für ein Experiment. In diesen Experimenten wurden verschiedene Versuche durchgeführt. Um schnell und effizient neue Modelle zu entwickeln und evaluieren, wurde eine neue Projektstruktur implementiert. Die neue Struktur ermöglichte, dass mit Parameteränderungen zwischen Datensätzen, Modellen und Klassifikation gewechselt werden konnte. Die Projektstruktur wird in Kapitel 9.2 genauer beschrieben.

5.2 Aufbau der Experimente

In diesem Abschnitt wird erklärt, welche Standardeinrichtungen in den Experimenten verwendet wurden. Sollte ein Experiment von dieser Einrichtung abweichen, wird im Experiment explizit darauf hingewiesen.

Preprocessing. Jeder Tweet wird vorverarbeitet. Dabei werden Tweets zuerst in Kleinbuchstaben umgewandelt und anschliessend die enthaltenen URLs und Benutzernamen mit einem standardisierten Token ersetzt. Hashtags werden in gewöhnliche Wörter transformiert. Die Tweets werden mit dem NLTK[4] TweetTokenizer in Token aufgeteilt. Anschliessend werden die Token mit einer ID ersetzt, welche auf die Vektorrepräsentation des Token zeigt.

Features. Als Features werden einzelne Tweets verwendet. Die Token in den Tweets werden mithilfe eines Vokabulars mit ihrer Vektorrepräsentation ersetzt. Die Vektorrepräsentation der Token $W \in \mathbb{R}^d$ werden anschliessend miteinander verkettet und bilden mit der Matrix $S \in \mathbb{R}^{n \times d}$ eine Repräsentation des Tweets. d steht für die Dimensionsgrösse der Embeddings und n für die maximale Anzahl der Token in einem Tweet. Um n zu bestimmen, wurde der längste Tweet in den Datensätzen betrachtet und dessen Länge auf die nächste Zehnerstelle aufgerundet. Nicht sinnvolle Tweets, wie zum Beispiel in nicht unterstützten Sprachen oder Tweets die ASCII-Art enthalten, wurden nicht betrachtet. Aufgrund dieser Analyse haben wir $n = 60$ gewählt. Tweets mit mehr als n Token werden auf die ersten n Token reduziert. Tweets, die weniger als n Token enthalten, werden mit Dummy-Token aufgefüllt. Token, die nicht im Vokabular vorhanden sind, werden mit Unknown-Token ersetzt. Standardmässig werden die Word-Embeddings mit der Dimension $d = 52$ verwendet.

[4] http://www.nltk.org

Ausgabeschicht. Für jedes Modell wurde eine Fully Connected-Schicht als Ausgabeschicht verwendet. Als Aktivierungsfunktion wird Softmax eingesetzt. Die Anzahl Ausgabeknoten sind abhängig von der Klassifizierungsaufgabe.

Optimization. Trainiert werden die Modelle mit dem AdaDelta-Optimizer. Der Hyperparameter $\epsilon = 10^{-5}$ wird eingesetzt und für die restlichen Parameter werden die Standardwerte verwendet.

Evaluierung. Die Modelle werden mittels *Stratified 10-Fold* Cross Validation evaluiert. Die Daten werden in einem 80:10:10 Verhältnis in die drei Kategorien eingeteilt: Training, Validation und Test. Im Gegensatz zum Training findet die Evaluation auf Autorenebene statt. Es wird die Leistung bei der Klassifizierung von Autoren betrachtet. Zur Klassifizierung eines Autoren werden die Vorhersagen der Tweets addiert. Die Klasse mit dem höchsten Wert wird zur vorhergesagten Klasse des jeweiligen Autoren.

Wenn man beispielsweise das Geschlecht eines Autoren anhand von drei Tweets $t_1, t_2.t_3$ ermitteln möchte, werden zuerst die Tweets einzeln klassifiziert, was in folgender Ausgabe resultieren könnte: $t_1 = [0.4, 0.6]$, $t_2 = [0.3, 0.7]$, $t_3 = [1.0, 0.0]$. Die erste Nummer jeder Ausgabe zeigt die Wahrscheinlichkeit, dass ein Tweet von einer Frau verfasst wurde und die zweite Nummer zeigt die Wahrscheinlichkeit, dass ein Tweet von einem Mann verfasst wurde. Die Ausgaben der Tweets t_1, t_2, t_3 werden anschliessend zusammengezählt was in folgender Ausgabe resultiert: $[1.7, 1.3]$. In diesem Beispiel, wäre der Autor als Frau klassifiziert worden.

Diese Resultate werden mit der Zero Rule (ZeroR) Klassifizierung verglichen. Die Genauigkeit von ZeroR auf einer Klassifizierung wird aus dem Verhältnis der am häufigsten vertretenen Klasse $max(K_f)$ zu allen Klassen $sum(K_f)$ berechnet:

$$baseline = \frac{max(K_f)}{sum(K_f)}. \tag{21}$$

Die Gleichung 21 zeigt die Berechnung der Baseline. K_f enthält die Häufigkeiten der Klassen in den Trainingsdaten. Die Accuracy-Baselines der englischen Daten finden sich in der Tabelle 7.

Klassifikation	Anzahl Klassen	PAN-2016	PAN-2017
Geschlecht	2	50,00 %	50,00 %
Sprachvariation	6	-	16,67 %
Alter	5	41,74 %	-

Tabelle 7: **Genauigkeit-Baselines in den englischen Datensätzen pro Klassifikation.**

Die Berechnung der ZeroR F1-Baseline erfolgt analog wie die der Accuracy-Baseline. Man berechnet den F1 Score, wenn nur die grösste Klasse vorhergesagt wird. Daraus resultieren die in Tabelle 8 ersichtlichen Werte.

Klassifikation	Anzahl Klassen	PAN-2016	PAN-2017
Geschlecht	2	33,33 %	33,33 %
Sprachvariation	6	-	4,76 %
Alter	5	11,78 %	-

Tabelle 8: **F1-Baselines in den englischen Datensätzen pro Klassifikation.**

Early Stopping. In jedem Fold des Cross Validation wird überprüft, ob sich ein Modell verbessert. Ist dies über 50 Epochen nicht mehr der Fall, wird dieser Fold beendet und das Training im nächsten Fold fortgesetzt.

Modell-Checkpoint. Während des Trainings werden die Resultate der einzelnen Epochen in einem Fold miteinander verglichen. Wenn der F1 Score eines Modells einen neuen Höchstwert erzielt, wird das Modell mit den trainierten Gewichten abgespeichert. Dieses Modell nennt sich *Modell-Checkpoint*.

Während des 10-Fold Cross Validation werden demzufolge 10 Modelle gespeichert.

Class Weights. Einige Modelle werden mit Class Weights trainiert. Class Weights werden nur für die Altersklassifizierung verwendet, weil der Datensatz für diese Aufgabe als einziger unausgeglichen ist. Falls in einen Experiment Class Weights zur Anwendung kommen, wird dies explizit erwähnt.

$$W = \frac{max(K_f)}{K_f} \tag{22}$$

Die Berechnung der Gewichte W ist in der Gleichung 22 ersichtlich. Der Vektor K_f enthält die Vorkommnisse aller Klassen in den Trainingsdaten. Die Matrix K_f wird durch die, mit der Max-Funktion gefundene, grösste Klasse dividiert.

Verwendete Daten. Die Experimente in dieser Arbeit werden mit englischen Tweets durchgeführt. Die Daten in den einzelnen Experimenten werden durch die vorherzusagende Klassen bestimmt.

Für die Vorhersage von *Geschlecht* und *Sprachvariation* eines Autoren, wird der PAN-2017-Datensatz verwendet und für die Vorhersage von *Alter* eines Autoren, der PAN-2016-Datensatz.

6 Experimente und Resultate

Die Beschreibung der Experimente und deren Resultate erfolgt in diesem Kapitel. In den Experimenten werden die Leistungen verschiedener Modelle ermittelt.

6.1 CNN-Modell

In diesem Experiment wird das Modell aus der Projektarbeit in die neue Projektstruktur integriert und dessen Resultate beschrieben. Durch die neue und übersichtlichere Projektstruktur wurde deutlich. dass im bestehenden Modell ein Fehler in der Evaluierung lag, was sich auf die Resultate ausgewirkt hatte. Dieser Fehler wurde behoben und das Modell erneuert trainiert. In den folgenden Abschnitten werden die Architektur und die Resultate beschrieben.

6.1.1 Architektur

Die Architektur dieses CNN-Modelles basiert auf der Arbeit von Deriu et. al. [2]. In diesem Abschnitt wird diese beschrieben und in der Abbildung 15 dargestellt.

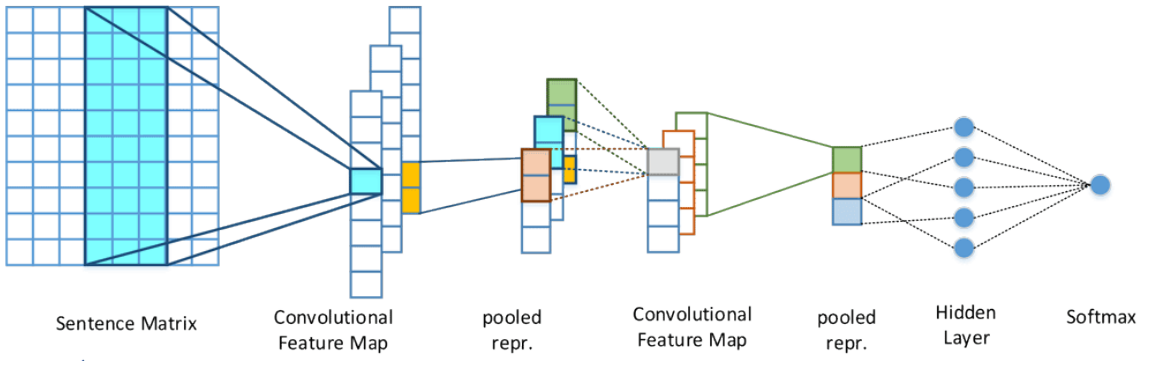

Abbildung 15: **Grafische Darstellung der Architektur des CNN-Modells [2].** Für eine übersichtliche Darstellung wurde $n = 4$ und $u = 3$ gewählt.

Zero Padding-Schicht. Eine Zero Padding-Schicht wird verwendet um die Eingabe für die Convolution-Schicht vorzubereiten. Die Eingabematrix S wird zwischen zwei leeren Matrizen mit der Dimension $\mathbb{R}^{h-1\times d}$ verkettet. Dies resultiert eine Matrix $S_p \in \mathbb{R}^{n+2(h-1)\times d}$. Zero-Padding wird eingesetzt, damit die ersten und letzten $h-1$ Token nicht vernachlässigt werden.

1. Convolution-Schicht. Die 1. Convolution-Schicht wird mit m Anzahl Filter und einem Window mit der Länge h verwendet. Für diese Schicht wurde $m = 200$ und $h = 6$ gewählt. Daraus resultiert eine Matrix $C_1 \in \mathbb{R}^{n+h-1\times m}$.

1. Max Pooling-Schicht. Die 1. Max Pooling-Schicht verwendet ein Pool mit der Länge s und dem Stride s_t. Für diese Schicht wurde $s = 4$ und $s_t = 2$ gewählt. Daraus resultiert eine Matrix $C_{p1} \in \mathbb{R}^{\frac{n+h-s}{s_1} \times m}$.

2. Convolution-Schicht. Die 2. Convolution-Schicht wird identisch zur 1. Convolution-Schicht verwendet. Daraus resultiert eine Matrix $C_2 \in \mathbb{R}^{\frac{n+h-s}{s_1} - h + 1 \times m}$.

2. Max Pooling-Schicht. Die 2. Max Pooling-Schicht wird verwendet um die gesamte Ausgabematrix der 2. Convolution-Schicht auf einen Vektor zu verkleinern. Um dies zu erreichen wird $s = \frac{n+h-s}{s_1} - h + 1$ und $s_t = 1$ gesetzt. Dadurch ist die Pool-Länge identisch mit der ersten Dimension von C_2 und daraus resultiert ein Vektor $C_{p2} \in \mathbb{R}^m$.

Hidden-Schicht. Nach der 2. Max Pooling-Schicht wird eine Hidden-Schicht mit m Neuronen eingesetzt. Als Aktivierungsfunktion wird relu verwendet. Daraus resultiert eine Matrix $h \in \mathbb{R}^m$.

6.1.2 Resultate

In diesem Abschnitt werden die Resultate des CNN-Modells beschrieben.

	Alter		**Geschlecht**	
	Genauigkeit	**F1 Score**	**Genauigkeit**	**F1 Score**
CNN	48,94 %	27,43 %	73,24 %	73,14 %
PAN-2016	38,79 %	-	55,75 %	-
Baseline	41,74 %	11,78%	50,00 %	33,33 %

Tabelle 9: **In dieser Tabelle sind die Resultate des CNN-Modells und der PAN-2016-Gewinner und die Baseline aufgelistet.** Die Resultate des CNN-Modells wurde mit Cross Validation bestimmt. Die PAN-2016-Gewinner wurden mit einem PAN-2016-Testset evaluiert.

In der Tabelle 9 ist ersichtlich, dass das CNN-Modell bereits gute Ergebnisse erzielt, besonders im Vergleich zur Baseline und den Gewinnern des PAN-2016-Wettbewerbs, letztere wurden jedoch auf einem anderen Datensatz getestet.

Das CNN-Modell macht instabile Vorhersagen. In der Abbildung 16 wird deutlich, dass die Genauigkeit je nach Datensatz stark schwankt. Eine Steigerung der Genauigkeit auf dem Validierungsset bedeutet nicht zwingend eine Steigerung auf dem Testset. Ein Beispiel dafür ist in der 37. Epoche ersichtlich. Während auf dem Validierungsset die zweithöchste Genauigkeit erzielt wird, wird auf dem Testset die zweittiefste Genauigkeit erzielt. Dies erschwert die Genauigkeit der Vorhersage des CNN-Modells auf unterschiedlichen Datensätzen.

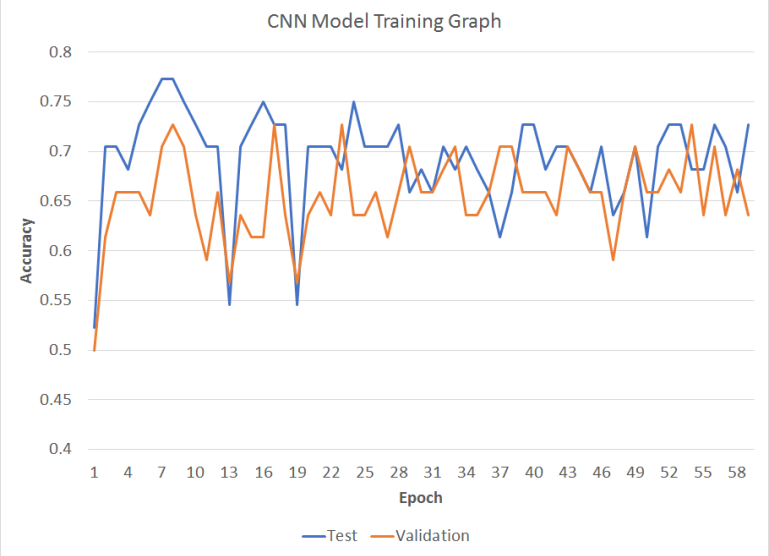

Abbildung 16: **Diese Grafik zeigt den Verlauf der Genauigkeit auf Validierungs- und Testdaten während des Trainings.**

6.2 CNN-LSTM-Modell

Dieses Kapitel beschäftigt sich mit der Frage, ob die Vorhersage von Alter und Geschlecht englischer Twitter-Nutzer verbessert werden kann, wenn zusätzliche Features eingesetzt werden. Das Experiment orientiert sich an der Arbeit von Sboev et al. [9], die ein ähnliches Experiment mit russischen Texten durchgeführt haben.

Die Arbeit von Rangel et al. [25] zeigt, dass die Verwendung zusätzlicher stylistischer Features einen positiven Einfluss auf die Author Proflierung haben kann.

In einem ersten Abschnitt werden die verwendeten Features besprochen. Anschliessend wird der Aufbau des verwendeten Modells erklärt. Abschliessend werden die Resultate beschrieben und diskutiert.

6.2.1 Features

In diesem Experiment wird der Einfluss von drei verschiedenen Variablen auf die Author Profiling Aufgabe untersucht: *Morphologische Features*, *Deriative Koeffizienten* und *Emotion Features*. In diesem Experiment werden die Word-Embeddings der Dimension $d = 200$ verwendet.

Morphologischen Features. Es wird für jeden Tweet analysiert, welche POS (Part of Speech) Eigenschaften enthalten sind. Basierend auf dieser Analyse werden für die einzelnen Tweets POS-Embeddings mit der Dimension $d = 12$ erstellt. In diesen Embeddings stehen die Häufigkeiten folgender Wortformen: Nomen, Zahlen, Adjektive, Präpositionen, Verben, Pronomen, Interjektionen, Adverbien, Artikel, Konjunktionen, Partizipien und Infinitive.

Die POS Tags wurden mit dem *Averaged Perceptron Tagger* von NLTK erstellt.

Psycholinguistische Merkmale. Für jeden Tweet werden anhand der POS Eigenschaften die folgenden Koeffizienten berechnet:

Der *Trager Koeffizient* CT ist das Verhältnis der verwendeter Verben VB zu den verwendeten Adjektiven ADJ. Der Trager Koeffizient sagt etwas aus über die emotionale Stabilität eines Menschen. Der Wert liegt normalerweise in der Nähe von 1.

$$CT = \frac{VB}{ADJ} \tag{23}$$

Der *Koeffizient: Bereitschaft zu Aktion* CRA ist das Verhältnis zwischen den verwendeten Verben VB zu den verwendeten Nomen NN. Dieser Koeffizient zeigt in welchem Ausmass ein Mensch sozialisiert ist. Der Wert liegt normalerweise ebenfalls in der Nähe von 1.

$$CRA = \frac{VB}{NN} \tag{24}$$

Der *Koeffizient der Agressivität* CA ist das Verhältnis zwischen den verwendeten Verben VB und deren Formen VBD, VBP, VBN, VBZ zu der Anzahl aller verwendeten Wörter (Token) T:

$$CA = \frac{VB + VBD + VBP + VBN + VBZ}{T}. \tag{25}$$

Wenn dieser Wert grösser als 0,6 wird, kann das ein Anzeichen für emotionale Unruhe eines Autoren sein.

Emotion Features. Es wird für jeden Tweet analyisiert, ob diese Rückschlüsse auf beteiligte Emotionen zulassen und wenn ja wie häufig diese vorkommen. Zusätzlich wird die Anzahl positiver und negativer Wörter (Sentiments) gezählt. Diese Informationen stammen aus dem in Kapitel 4.3 beschriebenen EmoLex. Anhand dieser Informationen werden Emotion-Embeddings mit der Dimension $d = 10$ erstellt. In diesen Embeddings sind die Häufigkeiten für die Emotionen: Wut, Angst, Erwartung, Vertrauen, Überaschung, Traurigkeit, Freude, Ekel, die Anzahl positiver und negativer Worte gespeichert.

6.2.2 Architektur

In diesem Abschnitt wird die Architektur des CNN-LSTM-Modells beschrieben. Dieses Modell entspricht - mit Ausnahme der Verwendung von Dropout - der Arbeit von Sboev et al. [9]. Wir haben uns in unserer Arbeit gegen diese entschieden, weil damit schlechtere Resultate erzielt wurden. In der Abbildung 17 wird die Architektur grafisch dargestellt.

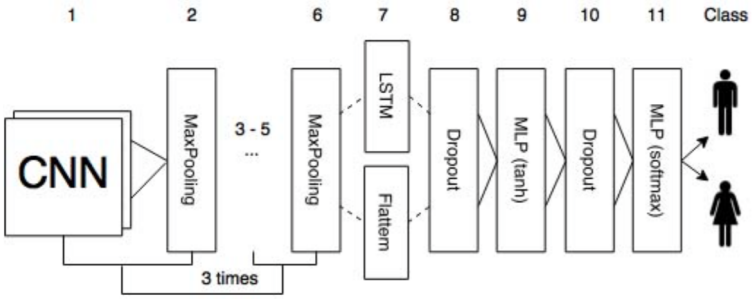

Abbildung 17: **Grafische Darstellung der Architektur des CNN-LSTM-Modells [9].** Die Anzahl der Ausgabeknoten in der 11. Schicht sind abhängig von der Klassifizierungsaufgabe.

Zero Padding-Schicht. Eine Zero Padding-Schicht wird verwendet, um die Eingabe für die Convolution-Schicht vorzubereiten. Daraus resultiert eine Matrix $S_p \in \mathbb{R}^{n+2(h-1)\times d}$.

Convolutional-Schicht. Die erste, dritte und fünfte Schicht sind CNN-Schichten. Diese werden jeweils mit einer Filteranzahl von $m = 30$ und einer Fenstergrösse der Länge $h = 3$ verwendet. Daraus resultiert in der Matrix $C_X \in \mathbb{R}^{n+h-1\times m}$ für $X \in (1, 2, 3)$.

Max Pooling-Schicht Die zweite, vierte und sechste Schicht sind Max Pooling-Schichten. Die Max Pooling-Schichten werden jeweils mit einem Pool der Länge $s = 2$ und dem Stride $s_t = 2$ verwendet. Daraus resultiert in der Matrix $C_{PX} \in \mathbb{R}^{n+h-1\times m}$ für $X \in (1, 2, 3)$.

LSTM-Schicht. Die siebte Schicht ist eine bidirektionale LSTM-Schicht mit u Einheiten. Die Ausgabematrix lautet $R \in \mathbb{R}^{2u\times n}$. Dieses Modell verwendet $u = 30$.

Optional: Derivative-Koeffizienten Schicht. Zwischen der siebten und achten Schicht wird in einer der Variationen eine zusätzliche Schicht eingesetzt. Diese Schicht berechnet, die in Kapitel 6.2.1 beschriebenen psycholinguistischen Merkmale. Die Ausgabe der LSTM-Schicht wird durch diese Schicht grösser und lautet $R \in \mathbb{R}^{2u\times n+3}$.

Tanh-Schicht. In der achten Schicht wird eine Fully Connected-Schicht und die Aktivierungsfunktion $tanh$ verwendet. Die Anzahl Ausgabeknoten beträgt $h = 10$.

Optimization. Trainiert wird dieses Modell mit dem Adam-Optimizer. Es werden die Standard Hyperparamater verwendet.

6.2.3 Variationen

Diese Architektur wurde unter Verwendung verschiedener Features in den fünf Variationen: *POS*, *POS+DERI*, *EMOTION*, *POS+EMOTION* und *POS+EMOTION+DERI* durchgeführt.

POS. In dieser Variation werden zusätzlich die in Kapitel 6.2.1 beschriebenen morphologischen Features verwendet.

POS+DERI. In dieser Variation wird zusätzlich zu den POS Features die in Kapitel 6.2.2 beschriebene *Derivative-Koeffizienten Schicht* eingesetzt.

EMOTION. In dieser Variation werden zusätzlich die in Kapitel 6.2.1 beschriebenen Emotion Features verwendet.

POS+EMOTION. In dieser Variation werden sowohl POS als auch Emotion Features verwendet.

POS+EMOTION+DERI. In dieser Variation wird zusätzlich zu den Emotion und POS Features die *Derivative-Koeffizienten Schicht* eingesetzt.

6.2.4 Resultate

In diesem Abschnitt werden die Resultate des CNN-LSTM-Modells und dessen Variationen beschrieben. In der Tabelle 10 werden die Resultate in der Alters- und Geschlechtsklassifizierung aufgelistet.

	Alter		**Geschlecht**	
(#) CNN-LSTM-Variante	**Genauigkeit**	**F1 Score**	**Genauigkeit**	**F1 Score**
(1) Default	49,61 %	27,34 %	78,56 %	78,53 %
(2) POS	48,26 %	26,65 %	78,22 %	78,18 %
(3) POS+DERI	48,07 %	27,33 %	78,61 %	78,83 %
(4) Emotion	50,122 %	28,49 %	78,61 %	78,53 %
(5) Emotion+POS	48,71 %	28,69 %	78,81 %	78,76 %
(6) Emotion+POS+DERI	50,116 %	28,27 %	78,72 %	78,67 %
Baseline	41,74 %	11,78 %	50,00 %	33,33 %

Tabelle 10: **Resultate der CNN-LSTM-Variationen auf englischen Tweets.**

Mit einer Genauigkeit von 50,122 % hat die Variante 4, das beste Ergebnis erreicht. Das Resultat übertrifft die Resultate des CNN-Modells damit um rund 1 %.

Das Geschlecht konnte mit keiner der Varianten signifikant besser erkannt werden als mit Variante 1. Im Vergleich zum CNN-Modell erzielt das CNN-LSTM-Modell in der Geschlechtserkennung um rund 5 % bessere Resultate.

Verwendung zusätzlicher Features und der Einfluss auf die Resultate. Die Arbeit von Sboev et al. [9] erzielt in russischer Sprache, mit der Verwendung von POS und Emotion Features eine Verbesserung von rund 2 % hinsichtlich der Genauigkeit der Geschlechtsklassifizierung. Diese Steigerung konnte in diesem Experiment in englischer Sprache nicht erreicht werden. Dafür konnte die Genauigkeit in der Altersklassifizierung unter Verwendung der Emotion Features im Vergleich zur Variante 1 um rund 1 % verbessert werden.

Die Verwendung der POS Features hat sowohl in der Geschlechts- als auch in der Alterserkennung zu schlechteren Resultaten geführt, als die Variante 1. Diese Tendenz zeigt sich auch in Variante 5, die schlechtere Resultate erzielt, als wenn nur Emotion Features verwendet werden. Wir erklären das mit den POS Tags des NLTK Taggers, die nicht für die in Tweets verwendeten Wörter optimiert sind.

Die Verwendung zusätzlicher psycholinguistischer Merkmale hat in der Variante 3 im Vergleich zur Variante 2 eine Steigerung des F1 Scores um rund 0,6 % erzielt. Dieser F1 Score fällt jedoch nicht höher aus, als in der Variante 1 ohne zusätzliche Features. Die Merkmale werden ebenfalls aus den POS Tags berechnet, dass diese nicht auf Tweets optimiert wurden zeigt sich auch in den Resultaten der Varianten, die zusätzlich die Derivative-Koeffizienten Schicht verwenden.

Vergleich der Genauigkeit auf Validierungs- und Testdaten. Abbildung 18 zeigt die Genauigkeit der Validierungs- und Testdaten während des Trainings.

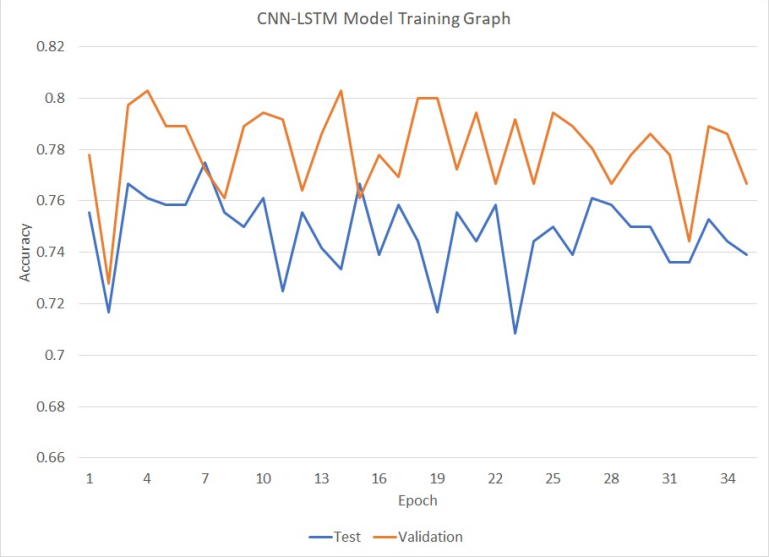

Abbildung 18: **Verlauf der Genauigkeit des CNN-LSTM-Modells auf Training und Testdaten.**

Wie man in Abbildung 18 sehen kann, weichen die beiden Graphen deutlich voneinander ab. Das Modell erzielt zwischen den Epochen 13 und 16 die höchste Genauigkeit auf den Validierungsdaten, zu diesem Zeitpunkt sind die Resultate auf den Testdaten jedoch, mit rund 3 %, weit vom möglichen Maximum entfernt.

6.3 Author-CNN-Modell

Ziel dieses Experiments war es, das CNN-Modell direkt mit Autoren zu trainieren statt auf einzelnen Tweets. Dabei wird zwischen einer Autoren- und Tweetebene unterschieden. Die Autorenebene ist gleichbedeutend mit gruppierten Tweets eines Autoren. Die vorherigen Experimente wurden auf der Tweetebene trainiert, was bedeutet, dass die Tweets keinem Autor zugeordnet und unabhängig voneinander trainiert werden. Das Modell soll mittels Autorenebene lernen, welche Tweets für eine Klassifizierung relevant sind. Beispielsweise enthalten nicht alle Tweets geschlechtsrelevante Informationen.

6.3.1 Architektur

Dieser Abschnitt zeigt, wie die Architektur des Author-CNN-Modells aufgebaut ist. Die Architektur wird in Abbildung 19 dargestellt. Die Architektur besteht aus zwei Ebenen: Einer Autorenebene und einer Tweetebene. Die Tweetebene ist bis und mit der 2. Max Pooling-Schicht identisch mit dem CNN-Modell und wird deshalb nicht genauer beschrieben.

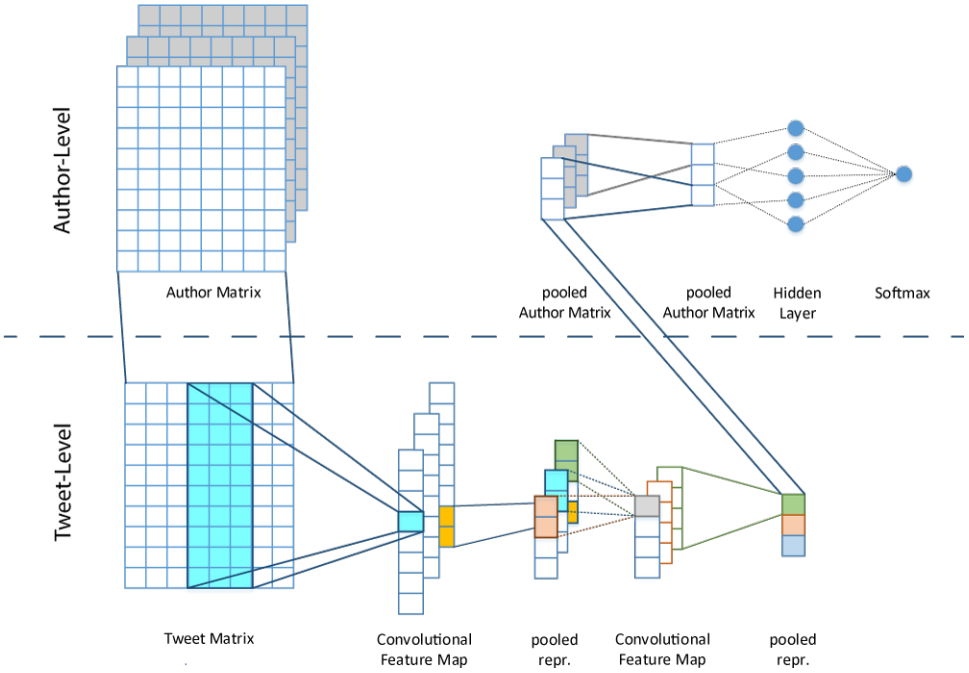

Abbildung 19: **Grafische Darstellung der Architektur des Author-CNN-Modells.** Ersichtlich ist die Unterscheidung zwischen der Autorenebene und Tweetebene. Die Autorenmatrix besteht aus mehreren Tweets und daraus werden einzelne Tweets der Tweetebene übergeben. Nach der Vereinfachung des Tweets wird dieser der Autorenebene zurückgegeben und den restlichen Tweets des Autoren zugeordnet.

Features. Dieses Modell trainiert nicht mehr mit einzelnen Tweets, sondern mit Autoren. Die einzelnen Tweets werden vorbereitet wie in Kapitel 5.2 erklärt, jedoch werden am Schluss die Tweets eines Autoren verkettet. Die neue Feature-Matrix $A \in \mathbb{R}^{a_n \times n \times d}$ (auch Autorenmatrix genannt) wird somit um eine Dimension erweitert. a_n steht für die maximale Anzahl Tweets pro Autor. Je nach Datensatz wurde a_n auf die maximale Anzahl Tweets eines Autoren gesetzt. Fehlende Tweets werden mit Nullen ergänzt.

Time Distributed-Schicht. Um dem CNN-Modell die einzelnen Tweets eines Autoren zu übergeben, wurde eine Time Distributed-Schicht eingesetzt. Die Tweets werden dadurch unabhängig vom Autor mit den identischen Gewichten trainiert. Die Ausgabe dieser Schicht ist die verkettete Ausgabe der Tweetebene und somit eine Matrix $A_p \in \mathbb{R}^{a_n \times m}$. m steht wie im CNN-Modell für die Anzahl Filter der CNN-Schichten.

Global Max Pooling-Schicht. Eine Global Max Pooling-Schicht dient der Vereinfachung der Tweets eines Autoren Die Matrix A_p, welche einer Liste von vereinfachten Tweets entspricht, wird zum Vektor $A_{p2} \in \mathbb{R}^m$ vereinfacht.

Hidden-Schicht. Diese Hidden-Schicht ist identisch mit der Hidden-Schicht im CNN-Modell. Sie besteht aus m Neuronen, verwendet Relu als Aktivierungsfunktion und es resultiert eine Matrix $h \in \mathbb{R}^m$.

6.3.2 Variationen.

In diesem Abschnitt wird die Author-CNN-Attention-Variation des CNN-Author-Modells beschrieben.

Author-CNN-Attention. Das Author-CNN-Attention-Modell (Author-CNN-ATT) verwendet eine Attention- anstelle einer Global Max Pooling-Schicht, mit dem Ziel alle Tweets eines Autoren zu vereinfachen. Die Ausgabematrix bleibt identisch.

6.3.3 Resultate

In diesem Abschnitt werden die Resultate des Author-CNN- und Author-CNN-ATT-Modells aufgeführt und mit dem CNN-Modell verglichen.

	Genauigkeit
CNN	73,24 %
Author-CNN	73,06 %
Author-CNN-ATT	71,79 %
Baseline	50,00 %

Tabelle 11: **Auflistung der Resultate der Author-CNN-Modelle.** In dieser Tabelle ist ein Vergleich der Genauigkeit verschiedener Modelle und der Baseline ersichtlich. Verglichen wird Geschlechtsklassifizierung.

In der Tabelle 11 werden die Modelle aufgrund der guten Verteilung der Daten in der Geschlechtsklassifizierung verglichen. Dadurch konnte der Fokus auf die Entwicklung des Modells gelegt werden. Beide Author-CNN-Modelle konnten die Resultate des CNN-Modells nicht übertreffen.

Die Tweetebene wird durch die zusätzliche Ebene vernachlässigt. Beim Trainieren beider Author-CNN-Varianten war zu sehen, dass die Modelle mit dem Training Schwierigkeiten hatte. Vermutlich wird dieses durch die zusätzliche Ebene erschwert. Durch die Vereinfachung von 100 Tweets auf einen Vektor der Länge 100 gehen viele Informationen verloren.

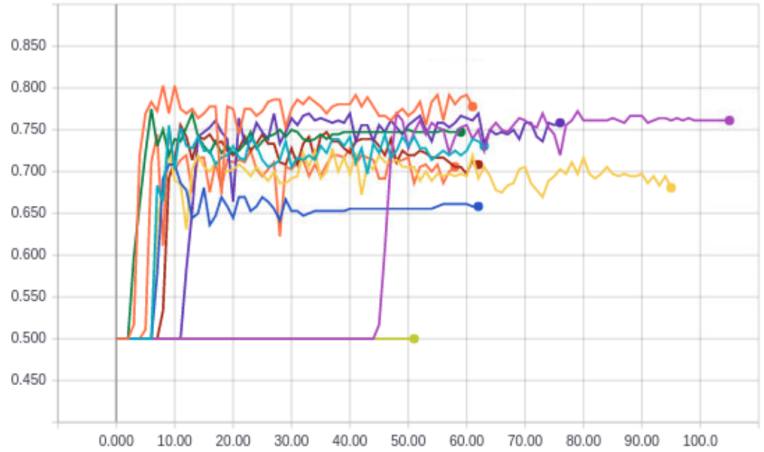

Abbildung 20: **Grafische Darstellung des Verlaufs der Genauigkeit während des Trainings des Author-CNN-ATT-Modells.** Die Genauigkeit der Geschlechtsklassifizierung auf dem Testset wird abgebildet. Die X-Achse steht für die Anzahl Epochen und die Y-Achse für die Genauigkeit. Die Linien repräsentieren die 10 Folds.

Abbildung 20 zeigt, dass der violette und hellgrüne Fold lange unverändert bleiben. Der violette Fold erzielt erst spät Fortschritte, der hellgrüne Fold bleibt 50 Epochen unverändert. Wir vermuten, dass durch die starke Vereinfachung mehrerer Tweets das Modell nicht mehr gut trainiert. Bei genauer Betrachtung der Attention-Schicht und deren Ausgabe fällt auf, dass die Tweets mehrheitlich gleich gewichtet werden. Das Ziel, den Score zu verbessern, indem auf Autorenebene trainiert wird, konnte in diesem Experiment nicht erreicht werden.

6.4 Hierarchical Network-Modell

Hierbei wurde ähnlich wie im vorherigen Experiment auf der Autorenebene trainiert mit dem Ziel dem Netzwerk beizubringen, welche Tweets für die Klassifizierung relevant sind.

6.4.1 Architektur

Für die Realisierung dieses Ziels, wurde ein Hierarchical Attention Network (HN-ATT) verwendet [4]. In diesem Abschnitt wird die Architektur des HN-ATT-Modells genauer beschrieben und in Abbildung 21 grafisch dargestellt.

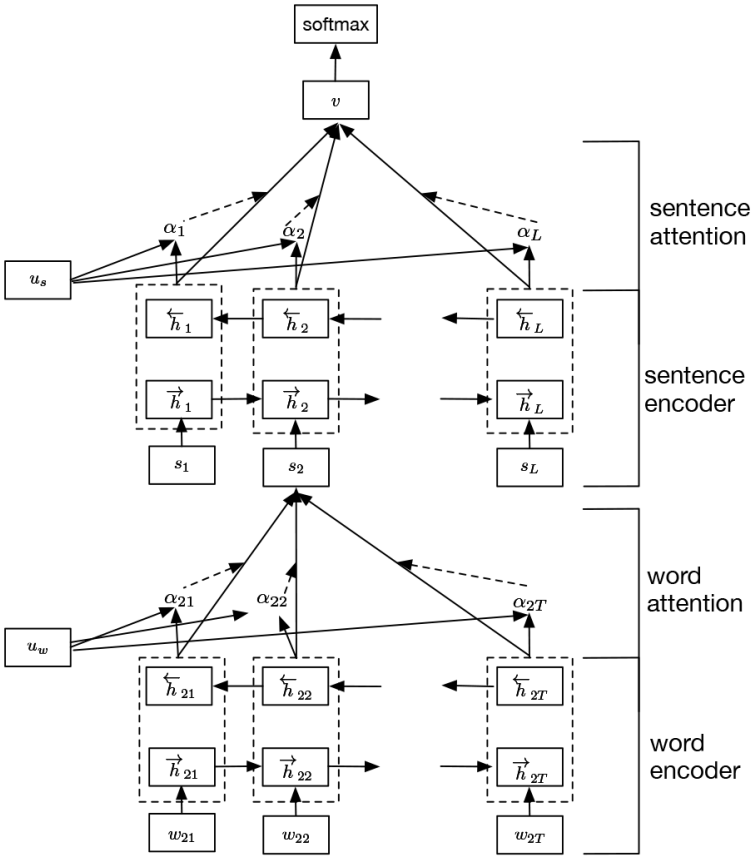

Abbildung 21: **Grafische Darstellung der Architektur des HN-ATT-Modells.** Word Encoder und Word Attention entsprechen der Tweetebene und Sentence Encoder und Sentence Attention entsprechen der Autorenebene. Die Token w_{2T} werden mit einer bi-GRU-Schicht encodiert und mit dem Attention-Mechanismus in Tweets s_L vereinfacht. Diese Tweets werden wiederum mit dem Attention-Mechanismus vereinfacht woraus eine Vereinfachung des Autoren v resultiert.

Features. Die Features in diesem Experiment sind identisch mit den Features des Author-CNN-Modells, welche im Kapitel 6.3.1 beschrieben werden.

Time Distributed-Schicht. Um encodierte Tweets zu erhalten müssen sie der Tweetebene übergeben werden. Die Ausgabe dieser Schicht ist die verkettete Ausgabe der Tweetebene. Die Schichten in der Tweetebene werden nachfolgend noch genauer beschrieben. Die Ausgabe dieser Schicht wird der GRU-Schicht der Autorenebene übergeben.

Tweetebene, GRU-Schicht. Diese Schicht ist eine bi-GRU-Schicht, die u Einheiten benötigen. Die Ausgabematrix ist $R \in \mathbb{R}^{2u \times n}$. Dieses Modell verwendet $u = 50$.

Tweetebene, Attention-Schicht. In dieser Schicht wird der Attention-Mechanismus auf Token angewendet. Dies resultiert ein Vektor $s_a \in \mathbb{R}^{2u}$. Der Vektor s_a wird der Time Distributed-Schicht übergeben.

Autorenebene, GRU-Schicht. Diese Schicht ist eine bi-GRU-Schicht und verwendet ebenfalls u Einheiten. Sie erhält die Ausgaben der Time Distributed-Schicht in der Matrix $A_p \in \mathbb{R}^{a_n \times u}$. Ausgegeben wird die Matrix $A_r \in \mathbb{R}^{a_n \times u}$.

Autorenebene, Attention-Schicht. Diese Schicht wendet den Attention-Mechanismus auf ganze Tweets an. Ausgegeben wird ein Vektor $A_{p2} \in \mathbb{R}^u$. In dieser Schicht soll die Relevanz eines Tweets gelernt werden.

6.4.2 Variationen

In diesem Abschnitt wird die HN-MAX-Variation des HN-ATT-Modells beschrieben.

HN-MAX. Anstelle von den Attention-Schichten wurden bei dieser Variante Global Max Pooling-Schichten eingesetzt. Die Ausgabematrizen bleiben identisch.

6.4.3 Resultate

In diesem Abschnitt werden die Resultate des HN-ATT- und HN-MAX-Modells beschrieben. In der Tabelle 12 sind die Resultate ersichtlich.

	Genauigkeit
CNN	73,24 %
HN-ATT	74,06 %
HN-MAX	73,75 %
Baseline	50,00 %

Tabelle 12: **Auflistung der Resultate der HN-Modelle.** In dieser Tabelle ist ein Vergleich der Genauigkeit verschiedener Modelle und der Baseline ersichtlich. Verglichen wurden die Resultate der Geschlechtsklassifizierung.

Tabelle 12 zeigt, dass beide Modelle das CNN-Modell leicht übertreffen. Im Anschluss werden weitere Analysen durchgeführt.

Die Reihenfolge der Tweets beeinflusst das Training. Gemäss der Arbeit von Yang et al. [4] sind Hierarchical Networks erfolgreich in der Lage, mehrere Sätze zu vereinfachen. In solchen Fällen spielt die Reihenfolge der einzelnen Sätze eine grosse Rolle. Tweets haben mehrheitlich keine Zusammenhänge und deshalb darf die Reihenfolge der Tweets bei der Klassifizierung keine Rolle spielen. Aus diesem Grund sind wir der Meinung, dass Hierarchical Networks eher ungeeignet für diesen Task sind.

Die Tweetebene wird vernachlässigt. Diese HN-Modelle haben im Vergleich zum CNN-Modell leicht bessere Resultate erreicht. Jedoch vermuten wir, dass sich die Schwierigkeiten des Author-CNN-Modell in diesem Experiment wiederholen. Aus diesem Grund wird im folgenden Experiment ein HN-ATT-Modell ohne Autorenebene verwendet.

6.5 GRU-Attention-Modell

In diesem Experiment wurde ein State of the Art Modell aus dem Natural Language Processing Gebiet auf das Author Profiling-Problem angewandt. Ein Modell, welches in den vergangen Jahren besonders viel Aufmerksamkeit erhielt, war das bi-GRU-Modell mit einem Attention Mechanismus (bi-GRU+Attention) von Bahdanau et al. [5]. Die im vorherigen Experiment beschriebenen Hierarchical Networks basieren ebenfalls auf der Arbeit von Bahdanau et al. Das bi-GRU+Attention-Modell hat in verschiedenen Bereichen wie Neural Machine Translation und Automatic Summarization neue Bestleistungen erzielt [26]. Deshalb wurde dieses Modell auf das Author Profiling-Problem angepasst und angewendet. Im Folgenden wird die Architektur des Modells erläutert und anschliessend die Resultate beschrieben.

6.5.1 Architektur

In diesem Abschnitt wird die Architektur des bi-GRU+Attention Modells beschrieben und in der Abbildung 22 grafisch dargestellt.

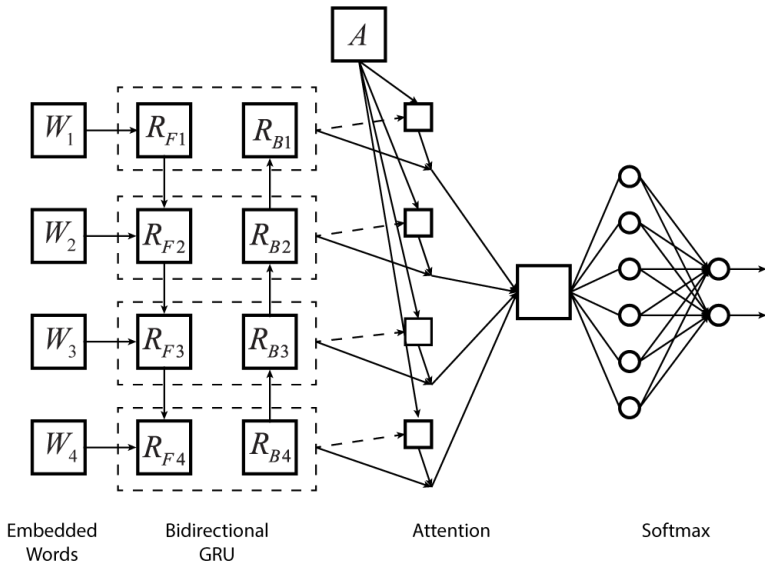

Abbildung 22: **Grafische Darstellung der Architektur des bi-GRU+Attention-Modells.** Für eine übersichtliche Darstellung wird $n = 4$ und $u = 3$ gewählt.

Masking-Schicht. Einer der Vorteile von RNNs sind die variable Eingabegrössen. Um variable Grössen zu unterstützen wird *Masking* angewendet. Masking funktioniert so, dass parallel zu den Daten auch eine Maske von Schicht zu Schicht weitergegeben wird. Die Maske entspricht einer Matrix, welche aus Einsen und Nullen besteht. Eine Null bedeutet, dass dieser Wert übersprungen werden soll und einen Eins, dass das Element nicht übersprungen werden darf. In dieser Schicht wird die Maske so berechnet, dass alle Unknown- und Dummy-Token übersprungen werden.

GRU-Schicht. Diese Schicht ist eine bi-GRU-Schicht, die u Einheiten benötigt. Ausgegeben wird die Matrix $R \in \mathbb{R}^{2u \times n}$. Dieses Modell verwendet $u = 50$.

Attention-Schicht. In dieser Schicht wird der Attention-Mechanismus implementiert. Dies resultiert einen Vektor $s_a \in \mathbb{R}^{2u}$, welcher eine Vereinfachung der ursprünglichen Matrix S ist.

Dropout. Dropout wird auf die Ausgabeschicht angewendet mit einer Drop-Rate von $p = 0.2$.

6.5.2 Variationen

bi-LSTM+Attention. Dieses Modell unterscheidet sich von bi-GRU+Attention dadurch, dass LSTM-Einheiten anstelle der GRU-Einheiten eingesetzt werden. Ansonsten ist dieses Modell im Aufbau identisch mit dem bi-GRU+Attention-Modell.

bi-GRU. In diesem Modell wird der Attention-Mechanismus nicht eingesetzt. Um die GRU-Schicht zu vereinfachen, wird nur das letzte Element der Ausgabesequenz verwendet. Somit hat die Ausgabe der GRU-Schicht dieselbe Form wie die Ausgabe der Attention-Schicht $s \in \mathbb{R}^{2u}$.

6.5.3 Resultate

In diesem Abschnitt werden die Resultate des bi-GRU+Attention Modells und dessen Variationen beschrieben.

	Alter		**Geschlecht**	
Varianten	**Genauigkeit**	**F1 Score**	**Genauigkeit**	**F1 Score**
bi-GRU+ATT	46,14 %	28,67 %	79,11 %	79,03 %
bi-LSTM+ATT	47,84 %	28,92 %	78,72 %	78,68 %
bi-GRU	48,98 %	29,5 %	79,33 %	79,26 %
Baseline	41,74 %	11,78%	50,00 %	33,33 %

Tabelle 13: **Auflistung der Resultate der GRU-Attention-Modelle.** In dieser Tabelle sind die Genauigkeiten der angegebenen Modelle für Geschlecht und Alter ersichtlich.

In der Tabelle 13 wird deutlich, dass die Resultate bezüglich Geschlechtsklassifizierung nahe beieinander liegen. Die guten Resultate des bi-GRU-Modells ohne Attention sind überraschend. Die Ursache, wieso der Attention-Mechanismus einen negativen Einfluss hat, konnte nicht gefunden werden.

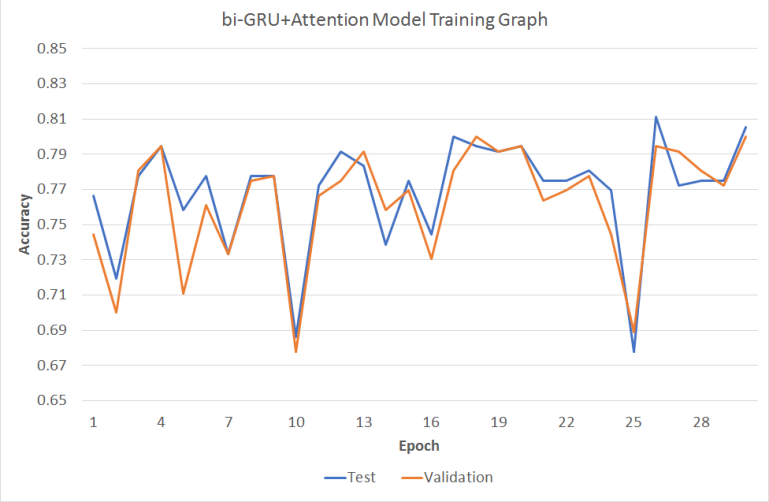

Abbildung 23: **Grafische Darstellung der Genauigkeit des bi-GRU+Attention-Modells während dem Training.** Die Genauigkeit wurde in der Geschlechtsklassifizierung auf dem Validation- und Test-Set gemessen.

RNNs machen sehr stabile Vorhersagen. In der Abbildung 23 ist ersichtlich, dass die Vorhersagen des bi-GRU+Attention-Modells auf verschiedenen Daten ähnlich sind. Ausschläge auf dem Validationset zeigen sich auch im Testset. Diese Eigenschaft macht das Modell auf neuen Daten vorhersehbar, was sich auch in Kapitel 6.7.2 auf den offiziellen Testdaten des PAN-Wettbewerbs gezeigt hat. Der Unterschied zwischen der mit Cross Validation erzielten Genauigkeit und der Genauigkeit auf den offiziellen Testdaten beträgt nur 0,23 %.

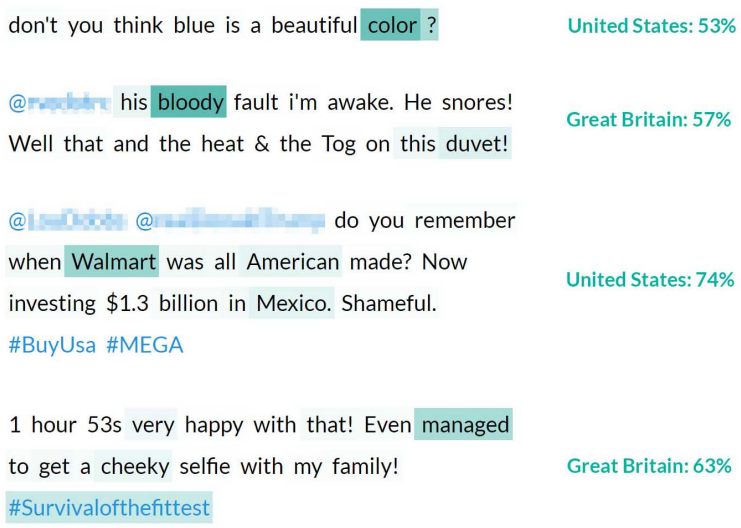

Abbildung 24: **Grafische Darstellung des Attention-Mechanismus für Language Variety Detection.** Auf der linken Seite ist die Gewichtung einzelner Wörter in einem Tweet zu sehen. Je dunkler die Farbe, desto stärker wurde das Wort gewichtet. Auf der rechten Seite ist die Vorhersage der Language Variety und deren Wahrscheinlichkeit ersichtlich. Alle Beispiel wurden korrekt vorhergesagt.

Attention auf Language Variety. Der Vorteil des Attention-Mechanismus liegt in der Analyse des Modells. Sie ermöglicht einen Einblick in das Modell ohne komplexe Berechnungen. Für eine Demonstration wurde ein Modell für die Erkennung der Sprachvariationen trainiert. Das Modell wurde auf englischen Daten trainiert und muss zwischen sechs Dialekten unterscheiden. Die Erkennung der Sprachvariationen wird verwendet, weil in dieser Aufgabe die Gewichtung der einzelnen Wörter am nachvollziehbarsten ist. Diese Resultate werden im Experiment 6.7.2 genauer beschrieben.

In der Abbildung 24 ist ersichtlich, wie der Attention-Mechanismus die einzelnen Wörter gewichtet. Das Modell erkennt, dass Wörter wie „color“ starke Indikatoren sind für amerikanisches Englisch. Zusätzlich hat das Modell gelernt, dass „Walmart“ sehr oft von Amerikanern verwendet wird. Das Modell ist auch in der Lage, britisch englische Wörter wie „bloody“ und „cheeky“ zu erkennen.

6.6 Experiment mit zusätzlicher Daten

Dieses Kapitel beschäftigt sich mit der Frage, ob die Altersklassifizierung englischer Twitter-Autoren unter Verwendung zusätzlicher Trainingsdaten verbessert werden kann. Dafür wurden die beiden Datensätze PAN-2016 und Hardiyan-2017 bearbeitet und kombiniert.

In einem ersten Abschnitt wird die verwendete Architektur des Experiments besprochen. Anschliessend dessen Variationen gezeigt und zum Abschluss die Resultate aufgezeigt.

6.6.1 Architektur

In diesem Experiment wurde das CNN-LSTM-Modell aus Experiment 6.2 in der Variante CNN-LSTM+Emotion verwendet. Dieses Modell hat in der Altersklassifizierung auf dem PAN-2016-Datensatz mit 50,12 % die höchste Genauigkeit erzielt.

6.6.2 Variationen

Die Durchführung des Experiments erfolgte mit unterschiedlichen Daten. Bei den neuen Datensätzen wurde darauf geachtet, dass ein Autor mindestens 70 Tweets verfasst hat. Es wurden ausserdem maximal 100 Tweets eines Autoren verwendet. Mit dieser Methode wurden die Datensätze *PAN16b*, *Hardiyan17b*, *PAN16b+Hardiyan17b*, *PAN16b+Hardiyan17b:Limited* und *PAN16+Hardiyan17:Limited* erstellt. Die für diese Variationen erstellten Datensätze werden im Anschluss genauer beschrieben.

PAN16b. In dieser Variante wurde der PAN-2016-Datensatz vor der Verwendung bearbeitet. Unter Anwendung der oben genannten Kriterien enthält der komprimierte Datensatz $N = 395$ Profile. Die Verteilung der Profile und Tweets über die einzelnen Alterskategorien ist in der Tabelle 14 ersichtlich.

Alterskategorie	18-24	25-34	35-49	60-64	65-XX	Total
Anzahl Profile	25	127	162	77	4	395
Anzahl Tweets	2490	12197	16023	4038	400	35148

Tabelle 14: **Verteilung der Profile und Tweets im PAN16b-Datensatz auf die fünf Alterskategorien.**

Es zeigt sich, dass sich die Verteilung der Profile nur minimal verändert hat. Die Tweets sind jedoch besser auf die einzelnen Profile verteilt.

Die Alterskategorie mit den meisten Profilen ist die Kategorie der 35-49 jährigen Autoren mit $k = 162$. In diesem Datensatz beträgt die Genauigkeits-Baseline:

$$ZeroR_{Genauigkeit} = \frac{162}{395} = 41,0\ \%. \tag{26}$$

Die F1-Baseline berechnet sich in diesem Datensatz aus $recall = 1$, $precision = \frac{162}{395} = 0,41$ und $n = 5$ Anzahl Klassen:

$$ZeroR_{F1} = \frac{2 * \frac{0,41*1}{0,41+1}}{5} = 23,89\ \%. \tag{27}$$

Der bearbeitete Datensatz weist eine um 0.74 % tiefere Genauigkeits-Baseline und eine um 4,56 % tiefere F1-Baseline auf als das Original.

Hardiyan17b. Bei dieser Variante wurde der Hardiyan17-Datensatz bearbeitet und verwendet. Unter Anwendung der oben genannten Kriterien enthält der komprimierte Datensatz $N = 3635$ Profile. Die Verteilung der Profile und Tweets über die einzelnen Alterskategorien ist in der Tabelle 15 aufgelistet.

Alterskategorie	18-24	25-34	35-49	60-64	65-XX	Total
Anzahl Profile	130	2125	1129	221	30	3635
Anzahl Tweets	12'244	209'282	111'281	21697	2982	357'486

Tabelle 15: **Verteilung der Profile und Tweets im Hardiyan17b-Datensatz auf die fünf Alterskategorien.**

Die Alterskategorie mit den meisten Profilen ist die Kategorie der 25-34-jährigen Autoren mit $k = 2125$. In diesem Datensatz beträgt die Genauigkeits-Baseline:

$$ZeroR_{Genauigkeit} = \frac{2125}{3635} = 58,46\ \%. \tag{28}$$

Die F1-Baseline berechnet sich in diesem Datensatz aus $recall = 1$, $precision = \frac{2125}{3635} = 0,58$ und $n = 5$ Anzahl Klassen:

$$ZeroR_{F1} = \frac{2 * \frac{0{,}58*1}{0{,}58+1}}{5} = 14,76\ \%. \tag{29}$$

Aufgrund der schlechten Verteilung der Datensätze resultieren hohe Baselines.

PAN16b+Hardiyan17b. Für diese Variante wurde der PAN16b- und der Hardiyan17b-Datensatz kombiniert. Daraus resultiert ein neuer Datensatz mit $N = 4030$ Profilen. Die Verteilung der Profile und Tweets über die einzelnen Alterskategorien wird in Tabelle 16 zusammengefasst.

Alterskategorie	**18-24**	**25-34**	**35-49**	**60-64**	**65-XX**	**Total**
Anzahl Profile	155	2252	1291	298	34	4030
Anzahl Tweets	14'734	221'879	127'304	29'335	3382	396'634

Tabelle 16: **Verteilung der Profile und Tweets im PAN16b+Hardiyan17b-Datensatz auf die fünf Alterskategorien.**

Die Alterskategorie mit den meisten Profilen ist die Kategorie der 25-34-jährigen Autoren mit $k = 2252$. In diesem Datensatz beträgt die Baseline:

$$ZeroR_{Genauigkeit} = \frac{2252}{4030} = 55,88\ \%. \tag{30}$$

Die F1-Baseline berechnet sich in diesem Datensatz aus $recall = 1$, $precision = \frac{2252}{4030} = 0,56$ und $n = 5$ Anzahl Klassen:

$$ZeroR_{F1} = \frac{2 * \frac{0{,}56*1}{0{,}56+1}}{5} = 14,34\ \%. \tag{31}$$

Auch in dieser Kombination zeigt sich die schlechte Verteilung des Hardiyan17b-Datensatzes. Die Verteilung der Daten wird in Abbildung 25 als Kreisdiagramm dargestellt.

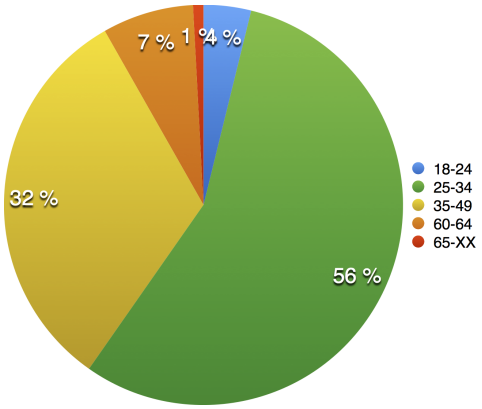

Abbildung 25: **Grafische Darstellung der Verteilung der Autoren im PAN16b+Hardiyan17b-Datensatz.**

Die schlechte Verteilung der Profile wird in dieser Grafik sehr deutlich. Aus diesem Grund wurde dieser Datensatz in einer folgenden Variante erneut bearbeitet.

PAN16b+Hardiyan17b:Limited. Hierfür wurde die Kombination aus PAN16b- und Hardiyan17b-Daten reduziert und darin eine maximale Anzahl von 240 Profile pro Alterskategorie definiert. Auf diese Weise verkleinerten sich die beiden Alterskategorien 25-34 und 35-49 und es resultiert ein neuer Datensatz mit $N = 876$ Profilen. Die Verteilung der Profile und Tweets auf die einzelnen Alterskategorien ist in Tabelle 17 ersichtlich.

Alterskategorie	**18-24**	**25-34**	**35-49**	**60-64**	**65-XX**	**Total**
Anzahl Profile	155	240	240	207	34	876
Anzahl Tweets	14'734	23'731	23'752	20'361	3382	85'960

Tabelle 17: **Verteilung der Profile und Tweets im PAN16b+Hardiyan17b:Limited-Datensatz auf die fünf Alterskategorien.**

In diesem Datensatz existieren zwei Kategorien mit der maximalen Anzahl von $k = 240$ Profilen. Daraus resultiert eine Baseline von:

$$ZeroR_{Genauigkeit} = \frac{240}{876} = 27,40\ \%. \tag{32}$$

Die F1-Baseline berechnet sich in diesem Datensatz aus $recall = 1$, $precision = \frac{240}{876} = 0,27$ und $n = 5$ Anzahl Klassen:

$$ZeroR_{F1} = \frac{2 * \frac{0,27*1}{0,27+1}}{5} = 8,60\ \%. \tag{33}$$

Durch eine Limitierung der Profile in einer Alterskategorie konnten beide Baselines in diesem Datensatz beinahe halbiert werden. Die Verteilung der Profile wird in Abbildung 26 als Kreisdiagramm dargestellt.

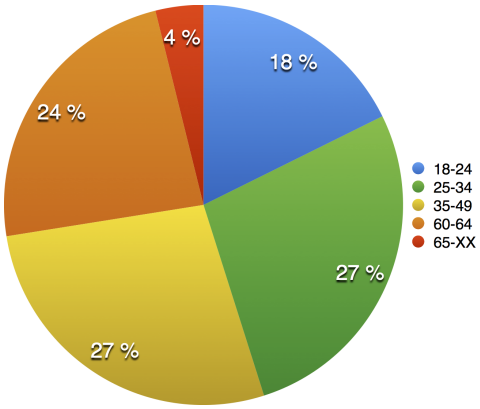

Abbildung 26: **Grafische Darstellung der Verteilung der Autoren im PAN16b+Hardiyan17b:Limited-Datensatz.**

Die Verteilung der Profile ist im Vergleich zum nicht limitierten Datensatz erheblich besser. Die Klasse der 18-24-Jährigen ist jetzt ähnlich gross wie die Baseline in diesem Datensatz. Der prozentuale Anteil der kleinsten Kategorie konnte von 1 % auf 4 % vergrössert werden.

PAN16+Hardiyan17:Limited. In dieser Variante wurde der PAN-2016-Datensatz unverändert übernommen. Dieser Datensatz wurde anschliessend mit Autoren aus dem Hardiyan17-Datensatz auf maximal 240 Autoren pro Alterskategorie ergänzt. Die Verteilung der Profile und Tweets über die einzelnen Alterskategorien ist in Tabelle 18 ersichtlich.

Alterskategorie	**18-24**	**25-34**	**35-49**	**60-64**	**65-XX**	**Total**
Anzahl Profile	240	240	240	228	55	1003
Anzahl Tweets	35'791	109'259	120'735	65'671	8249	907'525

Tabelle 18: **Verteilung der Profile und Tweets im PAN16+Hardiyan17:Limited-Datensatz auf die fünf Alterskategorien.**

In diesem Abschnitt existieren bereits drei Kategorien mit der maximalen Anzahl von $k = 240$ Profilen. Daraus resultiert eine Genauigkeits-Baseline von:

$$ZeroR_{Genauigkeit} = \frac{240}{1003} = 23,93\ \%. \tag{34}$$

Die F1-Baseline berechnet sich in diesem Datensatz aus $recall = 1$, $precision = \frac{240}{1003} = 0,24$ und $n = 5$ Anzahl Klassen:

$$ZeroR_{F1} = \frac{2 * \frac{0,27*1}{0,27+1}}{5} = 7,72\ \%. \tag{35}$$

Durch die Verwendung weiterer Profile konnte die Genauigkeits-Baseline in diesem Datensatz um 2,47 % und die F1-Baseline um 0,88 % reduziert werden. Die Verteilung der Profile wird in Abbildung 27 gezeigt.

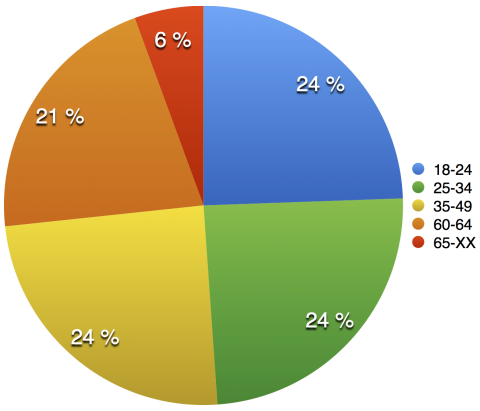

Abbildung 27: **Grafische Darstellung der Autoren im PAN16+Hardiyan17:Limited-Datensatz.**

Die Kategorie der über 65-Jährigen konnte um weitere 2 % auf Total 6 % vergrössert werden. Die Verteilung konnte somit noch ein wenig verbessert werden.

6.6.3 Resultate

In diesem Abschnitt werden die Resultate des CNN-LSTM+Emotion-Modells auf den zusätzlichen Daten beschrieben.

	Genauigkeit		F1 Score	
(#) Variante	**Baseline**	**Resultat**	**Baseline**	**Resultat**
(1) PAN16	41,74 %	50,12 %	11,78 %	28,49 %
(2) PAN16*	41,73 %	44,60 %	11,78 %	32,53 %
(3) PAN16b	41,00 %	41,69 %	11,63 %	23,89 %
(4) Hardiyan17b	58,46 %	59,92 %	14,76 %	20,24 %
(5) PAN16b+Hardiyan17b	55,88 %	57,99 %	14,34 %	21,95 %
(6) PAN16b+Hardiyan17b:Limited	27,40 %	41,45 %	8,60 %	32,66 %
(7) PAN16b+Hardiyan17b:Limited*	27,40 %	45,89 %	8,60 %	40,04 %
(8) PAN16+Hardiyan17:Limited	22,43 %	41,08 %	7,72 %	32,71 %
(9) PAN16+Hardiyan17:Limited*	22,43 %	43,47 %	7,72 %	38,09 %

Tabelle 19: **Resultate der Experimente mit zusätzlichen Daten in der Altersklassifizierung.** Alle Ergebnisse wurden mit dem CNN-LSTM+Emotion Modell erzielt. Die mit einem * markierten Varianten, wurden unter Verwendung von Class Weights durchgeführt.

In der Tabelle 19 ist ersichtlich, dass die Varianten 6 bis 9 in diesem Experiment im Vergleich zur Baseline bessere Resultate erzielt haben, als die Variante 1 mit dem Originaldatensatz. Bei den Varianten 3 bis 5 zeigte sich keine Verbesserung.

Mit dem originalen Datensatz in Variante 1 konnte eine Differenz von +8,38 % in Genauigkeit und einen Unterschied von +16,67 % in F1 Score im Vergleich zur Baseline erzielt werden.

Die grösste Differenz zwischen Genauigkeit und Baseline war in Variante 9 möglich. Die Genauigkeit von 43,47 % liegt dabei um 21,04 % höher als die dazugehörige Baseline. Der Unterschied zwischen dem erzielten F1 Score von 38,09 % zur Baseline liegt in dieser Variante bei +30,37 %.

Die grösste Verbesserung des F1 Scores im Vergleich zur Baseline wurde in Variante 7 gemessen. Die Differenz zwischen gemessenem F1 Score (40,04 %) und der Baseline liegt in dieser Variante bei +31,44 %. Der Unterschied zwischen Genauigkeit (43,47 %) und Baseline beträgt 18,49 %.

Mit mehr Daten können die einzelnen Klassen nicht automatisch besser klassifiziert werden. Eine Erhöhung der Datenmenge hat nicht automatisch positiven Einfluss auf die Klassifizierung. Wie in Variante 4 sichtbar wird, erhöhen sich bei der Verwendung schlecht verteilter zusätzlicher Daten nicht nur die Resultate, sondern auch die Baselines. Die Ergebnisse sind dann im Verhältnis zur Baseline sogar schlechter.

Eine gute Verteilung der Daten verbessert die Klassifizierung. Die beiden Varianten 6 und 8 weisen deutlich tiefere Baselines auf als die Variante 1 mit dem ursprünglichen Datensatz. Die Resultate liegen bei den beiden Varianten 6 und 8 sehr nah beieinander.

Die Genauigkeit in Variante 1 liegt mit 50,12 % rund 9 % höher als die Resultate in den beiden Varianten mit mehr Daten. Jedoch liegen die Baselines um 19,31 % in Variante 6 und 14,31 % in Variante 8 deutlich niedriger als die Baseline des ursprünglichen Datensatzes. In einem Vergleich zur Baseline erzielen damit beide Varianten deutlich bessere Ergebnisse nämlich +14,05 % in Variante 6 und +18,65 % in Variante 8.

Die erzielten F1 Scores bestätigen diese Feststellung. Die gemessenen F1 Scores liegen um mehr als 4 % höher als der F1 Scores auf dem PAN-2016-Datensatz. Bei Berücksichtigung der Baselines ergibt das eine Verbesserung von 7,39 % in Variante 6 und 8,32 % in Variante 8.

Mit beiden Evaluierungsmetriken konnte in den Varianten 6 und 8 eine Verbesserung festgestellt werden. Durch Betrachtung der Ergebnisse in Tabelle 19 lässt sich ein positiver Einfluss der Datenverteilung auf die Resultate erkennen.

Der Einsatz von Class Weights bei vielen Klassen hat positiven Einfluss auf die Resultate. In Variante 1 hat der Einsatz von Class Weights eine geringere Genauigkeit zur Folge. Jedoch erhält man eine Verbesserung des F1 Scores von rund 4 %.

In den beiden Varianten mit mehr Daten und besserer Verteilung erreicht man sogar eine Verbesserung des F1 Scores um 7,38 % in Variante 7 beziehungsweise 5,38 % in Variante 9. Diese Verbesserung ist in diesen Datensätzen aufgrund der besseren Verteilung auch bei der Genauigkeit messbar.

In allen drei Versuchen konnte durch die Verwendung von Class Weights eine Verbesserung des F1 Scores gemessen werden. Bei genügend grossen und gleichmässig verteilten Datensätzen hat der Einsatz von Class Weights sogar positiven Einfluss auf die Genauigkeit.

6.7 PAN-2017-Wettbewerb

In diesem Abschnitt wird die Teilnahme am PAN-2017-Wettbewerb beschrieben. Die Shared Tasks in diesem Jahr enthalten die Klassifizierung von Geschlecht und Sprachvariation in den vier Sprachen: Englisch, Spanisch, Portugiesisch und Arabisch. Der PAN-Wettbewerb selbst wird in Kapitel 9.1 genauer beschrieben.

Nachstehend wird zuerst erläutert wie das Modell für die Teilnahme am Wettbewerb ausgewählt wurde. Anschliessend werden die Resultate mit den offiziellen Testdaten des Wettbewerbs aufgelistet und besprochen.

6.7.1 Modell Auswahl

In diesem Experiment wurden zwei Modelle miteinander verglichen. Das bi-GRU+Attention-Modell aus Kapitel 6.5, welches zum Zeitpunkt dieses Experiments die besten Resultate in der Erkennung von Geschlecht erzielte und das CNN-Modell. Mittels CNN-Modell aus Kapitel 6.1 wurde die Verbesserung in der Geschlechtserkennung gemessen.

Beide Modelle wurden mit Cross Validation sowohl auf die Erkennung des Geschlechts als auch auf die Erkennung der Sprachvariation mit englischen Daten trainiert. Die Resultate sind in Tabelle 20 zusammengefasst.

Modell	Geschlecht	Anzahl Sprachvariationen	Sprachvariation
bi-GRU+Attention	79,11 %	6	79,03 %
CNN	73,24 %	6	70,90 %

Tabelle 20: **Auflistung der Resultate in der Erkennung von Geschlecht und Sprachvariation englischer Autoren.** Die erzielten Resultate wurden auf den englischen Tweets des PAN-2017-Datensatzes erzielt.

Während des Trainings erkannte das bi-GRU+Attention-Modell das Geschlecht eines Autoren mit einer Genauigkeit von 79,11 % und die Sprachvariation mit einer Genauigkeit von 79,03 %. Dieses Modell erzielte im Vergleich zum CNN-Modell um 5,87 % bessere Resultate in der Erkennung des Geschlechts und um 8,13 % in der Erkennung der Sprachvariation eines Autoren.

Aufgrund dieser Resultate wurde das bi-GRU+Attention-Modell für die Teilnahme am Wettbewerb ausgewählt. Die Resultate des Wettbewerbs werden im nächsten Abschnitt besprochen.

6.7.2 Resultate

In diesem Abschnitt werden die im PAN-2017 erzielten Resultate des bi-GRU+Attention-Modells anhand der Tabelle 21 aufgezeigt und besprochen.

Sprache	Joint	Geschlecht	Anzahl Sprachvariationen	Sprachvariation
Englisch	62,63 %	78,88 %	6	79,98 %
Spanisch	66,46 %	72,17 %	7	91,43 %
Portugiesisch	73,00 %	78,13 %	2	93,50 %
Arabisch	56,88 %	71,50 %	4	76,88 %

Tabelle 21: **Auflistung der Genauigkeit der offiziellen Resultate in der Erkennung von Geschlecht und Sprachvariation auf verschiedenen Sprachen.** Diese Resultate wurden mit dem bi-GRU+Attention-Modell erzielt.

Der höchste Wert in der Klassifizierung des Geschlechts wurde mit 78,88 % in Englisch erzielt. Portugiesisch liegt mit einem Unterschied von 0,075 % nur leicht unter diesem Wert. Die Geschlechtsklassifizierungen in Spanisch und Arabisch liegen jedoch deutlich unter diesen Ergebnissen.

In der Erkennung der Sprachvariation wurden ohne grosse Anpassungen des Modells gute Resultate erzielt. Hervorzuheben ist mit 91,43 % das Resultat in der Erkennung der Sprachvariation in Spanisch, die aus sieben verschiedenen Sprachvariationen besteht. Nur in den Sprachen Englisch und Arabisch liegen die erzielten Genauigkeiten unter 80 %.

Abdeckung des Vokabulars und der Einfluss auf die Klassifizierung des Geschlechts. Vor Verwendung der Word-Embeddings wurde anhand der PAN-2017-Trainingsdaten überprüft, wie hoch die Abdeckung der verwendeten Wörter ist. Diese Abdeckung ist wichtig, weil unbekannte Wörter mit einem standardisierten Token ersetzt werden. Die Abdeckung in den verwendeten Word-Embeddings ist in Tabelle 22 ersichtlich.

Sprache	Geschlecht Genauigkeit	Abdeckung des Vokabulars
Englisch	78,88 %	90,85 %
Portugiesisch	78,13 %	88,33 %
Spanisch	72,17 %	79,68 %
Arabisch	71,50 %	77,66 %

Tabelle 22: **Auflistung der Vokabularabdeckung im Vergleich zu den erzielten Genauigkeiten in der Klassifizierung des Geschlechts.**

Die Resultate in Tabelle 22 deuten an, dass die Genauigkeit in der Erkennung des Geschlechts eines Autoren mit der Abdeckung des Vokabulars in einem Datensatz zusammenhängen.

In der Erkennung von Sprachvariationen ist dieser Effekt nicht aufgetreten. Hier sind typische Wörter der Sprachvariationen stärker gewichtet als eine möglichst grosse Abdeckung der Wörter in den Word-Embeddings. Zum Beispiel wird das Englische Wort „color“ im britischen Englisch „colour“ geschrieben. Diese sprachlichen Unterschiede zwischen den Sprachvariationen sind für diese Klassifikation wichtig. Wörter, die nur von wenigen Autoren verwendet werden, sind üblicherweise nicht entscheidend.

Resultate in der Klassifizierung der Sprachvariation im Vergleich zu den Baselines. In diesem Abschnitt werden die Resultate in der Klassifikation der Sprachvariationen mit den jeweiligen Baselines verglichen. In Tabelle 23 sind die Resultate, die Baseline der jeweiligen Sprache und die Differenz zwischen erzielter Genauigkeit und Baseline aufgelistet.

Sprache	Baseline	Genauigkeit	Differenz
Englisch	16.67 %	79,08 %	62,41 %
Spanisch	14,29 %	91,43 %	77,14 %
Portugiesisch	50,00 %	93,50%	43,50 %
Arabisch	25,00 %	76,88 %	51,88 %

Tabelle 23: **Auflistung der erzielten Genauigkeit in der Klassifizierung der Sprachvariationen im Vergleich zur Baseline.**

Wie Tabelle 23 zeigt, sind die Baselines in Portugiesisch und Arabisch deutlich höher als die Baselines in Englisch und Spanisch. Die Baselines hängen von der Anzahl möglicher Sprachvariationen N in einer Klasse: $Baseline = \frac{100\ \%}{N}$ ab. Aufgrund der berechneten Differenzen zwischen Baseline und erzielter Genauigkeit können die einzelnen Resultate fair miteinander verglichen werden.

Alle Resultate liegen weit über der Baseline der jeweiligen Sprache. Mit einer Differenz von 77,14 % erzielt das Modell in der Erkennung der Sprachvariation in Spanisch das mit Abstand beste Resultat.

7 Schlussfolgerungen

In unseren Experimenten in Kapitel 6 haben wir Modelle für das Author Profiling-Problem entwickelt, getestet und analysiert. Die besten Resultate dieser Experimente werden in den Tabellen 24 bis 26 zusammengefasst.

Experiment	Genauigkeit	F1 Score
CNN	73,24 %	73,14 %
CNN-LSTM	78,81 %	78,83 %
Author-CNN	73,06 %	-
HN-ATT	73,75 %	-
GRU	79,33 %	79,26 %
Baseline	50,00 %	33,33 %

Tabelle 24: **Höchste Resultate der Experimente in der Geschlechtserkennung auf englischen Tweets.**

Tabelle 24 zeigen den Fortschritt in der Erkennung des Geschlechts auf dem PAN-2017-Datensatz. Die Resultate des CNN-Modells konnten mit unseren Modellen um bis zu 6 % übertroffen werden.

Experiment	Genauigkeit	F1 Score
CNN	48,94 %	27,43 %
CNN-LSTM	50,12 %	28,69 %
GRU	48,98 %	29,50 %
Baseline	41,74 %	11,78 %

Tabelle 25: **Höchste Resultate der Experimente in der Alterserkennung auf englischen Tweets.**

In der Tabelle 25 ist ersichtlich, dass auf dem PAN-2016-Datensatz eine Verbesserung von rund 1 % hinsichtlich Genauigkeit und rund 2 % im F1 Score in der Alterserkennung erzielt wurde. Die Genauigkeit des GRU-Modells entspricht mit 49,98 % praktisch dem Ergebnis des CNN-Modells, es erzielt jedoch einen deutlich höheren F1 Score. Die Vorhersagen sind entsprechend besser über die einzelnen Klassen verteilt.

Experiment	Genauigkeit	F1 Score
CNN	70,92 %	71,11 %
GRU	79,08 %	79,16 %
Baseline	16,67 %	4,76 %

Tabelle 26: **Höchste Resultate der Experimente in der Erkennung der Sprachvariation auf englischen Tweets.**

Tabelle 26 listet die Resultate in der Erkennung der Sprachvariation auf. Das GRU-Modell übertrifft mit einer Genauigkeit von 79,08 % die Resultate des CNN-Modells um rund 8 %.

Das GRU-Modell liefert in allen Aufgaben die höchsten F1 Scores. Einzig in der Erkennung des Alters erreicht das CNN-LSTM-Modell eine höhere Genauigkeit. Die Resultate in F1 Score sind deutlich aussagekräftiger, weil hier auch die Verteilung der Daten Berücksichtigung findet.

Im Experiment mit erweiterten Datensätzen in Kapitel 6.6 hat sich gezeigt, dass die Ergebnisse der F1 Scores deutlich gesteigert werden können, wenn die Daten in den einzelnen Klassen gleichmässig verteilt sind. Das beste Resultat mit diesen Datensätzen übertrifft die F1-Baseline um 30,37 %. Dies entspricht beinahe einer Verdoppelung der Differenz des besten F1 Scores auf dem PAN-2016-Datensatz im Vergleich zur Baseline.

In dieser Arbeit wurde gezeigt, dass die Resultate des PAN-16-Wettbewerbs mit Cross Validation übertroffen werden können. Dies wurde im Vergleich mit dem CNN-Modell festgestellt, das in ähnlicher Form an diesem Wettbewerb teilgenommen hat. Die Genauigkeit bezüglich Alterserkennung konnte um rund 1 % und in der Geschlechtserkennung um rund 6 % verbessert werden.

Das GRU-Modell erzielte gute Resultate in allen in dieser Arbeit betrachteten Klassifizierungsaufgaben und Sprachen. Das Ziel dieser Arbeit, ein Modell zu entwickeln, das sowohl auf verschiedenen Sprachen als auch auf verschiedenen Aufgaben funktioniert, ist somit erreicht worden.

8 Ausblick

In diesem Kapitel werden weiterführende Überlegungen zusammengefasst und offene Fragestellungen skizziert.

CNN-LSTM-Experiment. Diese Arbeit musste sich auf eine Auswahl der in der Arbeit von Sboev et al. [9] erwähnten Features beschränken. Der Einfluss syntaktischer Features auf die Alters- und Geschlechtserkennung fehlt.

Auch die Experimente mit POS Features und psychologischen Merkmalen müssten unter Verwendung von Tweet optimierten POS Tags wiederholt werden.

Der Einsatz von Emotion Features lieferte sowohl in der Arbeit von Rangel et al. [25] als auch in dieser Arbeit vielversprechende Resultate. Es müsste analysiert werden, ob dieser Effekt durch die Verwendung eines grösseren Lexikon weiter verstärkt werden könnte.

Author-CNN-Experiment. Möglicherweise gehen durch das Training auf Autorenebene Informationen verloren. Dieses Problem könnte eventuell gelöst werden, indem das Modell zuerst auf Tweetebene und im Anschluss auf beiden Ebenen trainiert wird.

Hierarchical-Attention-Experiment. Ein Einfluss der Reihenfolge der Tweets auf das Training dieses Modells wäre denkbar. Um dies zu vermeiden und dennoch ein Training auf einzelnen Tweets zu ermöglichen, könnte die Reihenfolge der Tweets in jeder Epoche zufällig verändert werden.

Daten. Die Verteilung der Daten war selbst mit der Verwendung zusätzlicher Daten noch nicht optimal. Die Resultate bei exakt gleich grossen Klassen und bei Datensätzen mit deutlich mehr Autoren müssten noch untersucht werden.

Die Arbeit von Radford et al. [27] löst das Problem kleiner Datensätze mit der zusätzlichen Verwendung von Unsupervised-Daten. So muss nur nach Tweets in der richtigen Sprache gesucht werden, um die Datenmenge erheblich zu erhöhen. Dieser Ansatz bedingt jedoch auch eine deutlich längere Trainingsphase. Das Modell in der Arbeit von Radford et al. [27] musste einen Monat lang auf vier GPUs trainiert werden.

Features. Es könnten spezifische Informationen zu den einzelnen Aufgaben hinzugenommen werden, zum Beispiel die Verwendung von geschlechtsspezifischen Worthäufigkeiten bei der Geschlechtsklassifizierung oder typischer Wörter einer Sprachvariation bei deren Klassifizierung.

Schliesslich könnte man die Modelle nicht nur auf Wörter sondern auch auf Charakterebene trainieren. In der Arbeit von Chiu et al. [28] wird mit dieser Methode State of the Art Performanz in Named Entity Recognition erzielt.

9 Anhang

9.1 PAN

PAN[5] ist eine Serie verschiedener digitaler Textanalyseaufgaben. Die Serie beinhaltet Aufgaben in den drei Themen: *Authorship*, *Orginality* und *Trust*. Der Fokus dieser Arbeit liegt in einer der Authorship-Aufgaben. Authorship beinhaltet die drei *Shared Tasks* genannten Evaluierungsaufgaben: *Author Identification*, *Author Profiling* und *Author Obfuscation*.

Im Rahmen dieser Arbeit beschäftigen wir uns ausschliesslich mit der Author Profiling-Aufgabe.

Author Profiling. Beim Author Profiling Task liegt der Fokus auf der Erkennung bestimmter demographischer Eigenschaften eines Autoren. Diese Aufgabe wird in einem ähnlichen Umfang bereits seit 2013 angeboten. Im Jahr 2016 haben die beiden Aufgaben dieses Tasks die Erkennung von Alter und Geschlecht eines Autoren auf Twitter beinhaltet. Im PAN-16 wurden die drei Sprachen: Englisch, Spanisch und Holländisch unterstützt.

Im Jahr 2017 wurde die Erkennung des Alters durch die Erkennung von Sprachvariationen ersetzt. Diese Aufgaben lagen in den vier Sprachen: Englisch, Spanisch, Portugiesisch und Arabisch vor.

TIRA. Im Rahmen des Wettbewerbs wurden die trainierten Modelle selbst ausgewertet. Dies war über die Evaluation-as-a-Service Plattform *TIRA*[6] möglich. Die offiziellen Testdaten waren nur über diese Plattform verfügbar und konnten nicht durch die Teilnehmer eingesehen werden.

Evaluierung. Die Evaluierung der Modelle geschieht im PAN-Wettbewerb über die Genauigkeit. Diese Metrik wird in Kapitel 3.5 genauer erklärt. Es wird in jeder Sprache für jede Teilaufgabe die Genauigkeit der Vorhersage berechnet. Zusätzlich wurde die Genauigkeit berechnet, wenn beide Vorhersagen (Geschlecht und Sprachvariation) korrekt waren. Dieser Score wird *JOINT* genannt. Die finale Rangierung geschieht für alle Sprachen über die durchschnittliche Genauigkeit über die erzielten Genauigkeiten.

[5] http://www.pan.webis.de
[6] http://www.tira.io

9.2 Projektstruktur

Dieses Kapitel bietet einen groben Überblick über die verwendete Projektstruktur. Diese Arbeit setzt folgende Technologien ein:

- Python 3.5.3 [7]
- Docker 17.03.1-ce [8],
- NVIDIA Docker 1.0.0 [9]
- TensorFlow 1.1.0-py3 [29]
- NumPy 1.11.1 [10]

Der Fokus liegt auf die Abstraktion der einzelnen Komponenten um mühelos zwischen Modellen, Datensätzen, Klassifizierungen und verschiedenen Arten von Features wechseln zu können. In der Abbildung 28 ist ein Klassendiagramm der wichtigsten Komponenten ersichtlich. Im Anschluss werden diese genauer beschrieben.

Task. Die abstrakte Methode `run` der `Task`-Klasse ist der Startpunkt der Applikation. Von dieser Klasse wird geerbt und die Methode `run` implementiert um eine neue Aufgabe zu erstellen. Durch die Angabe von `task_name` beim Start der Applikation wird der Task initialisiert und die `run`-Methode ausgeführt. Die `ExperimentTask`-Klasse ist eine Unterklasse von `Task` und enthält Hilfsfunktionen, um Modelle zu evaluieren, zu trainieren und zu testen. Die Experimente erben von der `ExperimentTask`-Klasse und die genauen Aufgaben werden in der `run` Methode implementiert.

Model. Die `Model`-Klasse enthält Funktionalitäten, die alle Modelle benötigen. Die zwei Klassen `AuthorModel` und `TweetModel` erben von `Model` und unterscheiden sich durch die Art der Klassifizierung von Autoren. Je nach Modell erbt die Implementation von der `AuthorModel`- oder `TweetModel`-Klasse. In der `_init_model`-Methode muss das Modell implementiert werden. Für die Implementierung der Modelle muss die Keras-API von TensorFlow verwendet werden.

Embedding. Die `Embedding`-Klasse ist eine abstrakte Klasse und wird für die Implementation verschiedener Word Embedding eingesetzt. Sie enthält eine Instanz der `Vocabulary`-Klasse und die Gewichte in einem Numpy-Array.

DatasetLoader. Die `DatasetLoader`-Klasse ist zuständig um die Daten zu laden und in eine Liste mit Instanzen der `Author`-Klasse abzufüllen. In der `load`-Methode ist eine Cache-Funktionalität implementiert. Dadurch wird der gesamte Datensatz verarbeitet und zwischengespeichert. Zusätzlich wird damit der Entwicklungsvorgang stark beschleunigt. Dies muss jedoch beachtet werden, wenn der Datensatz verändert wird.

[7] https://www.python.org/
[8] https://www.docker.com/
[9] https://github.com/NVIDIA/nvidia-docker
[10] http://www.numpy.org/

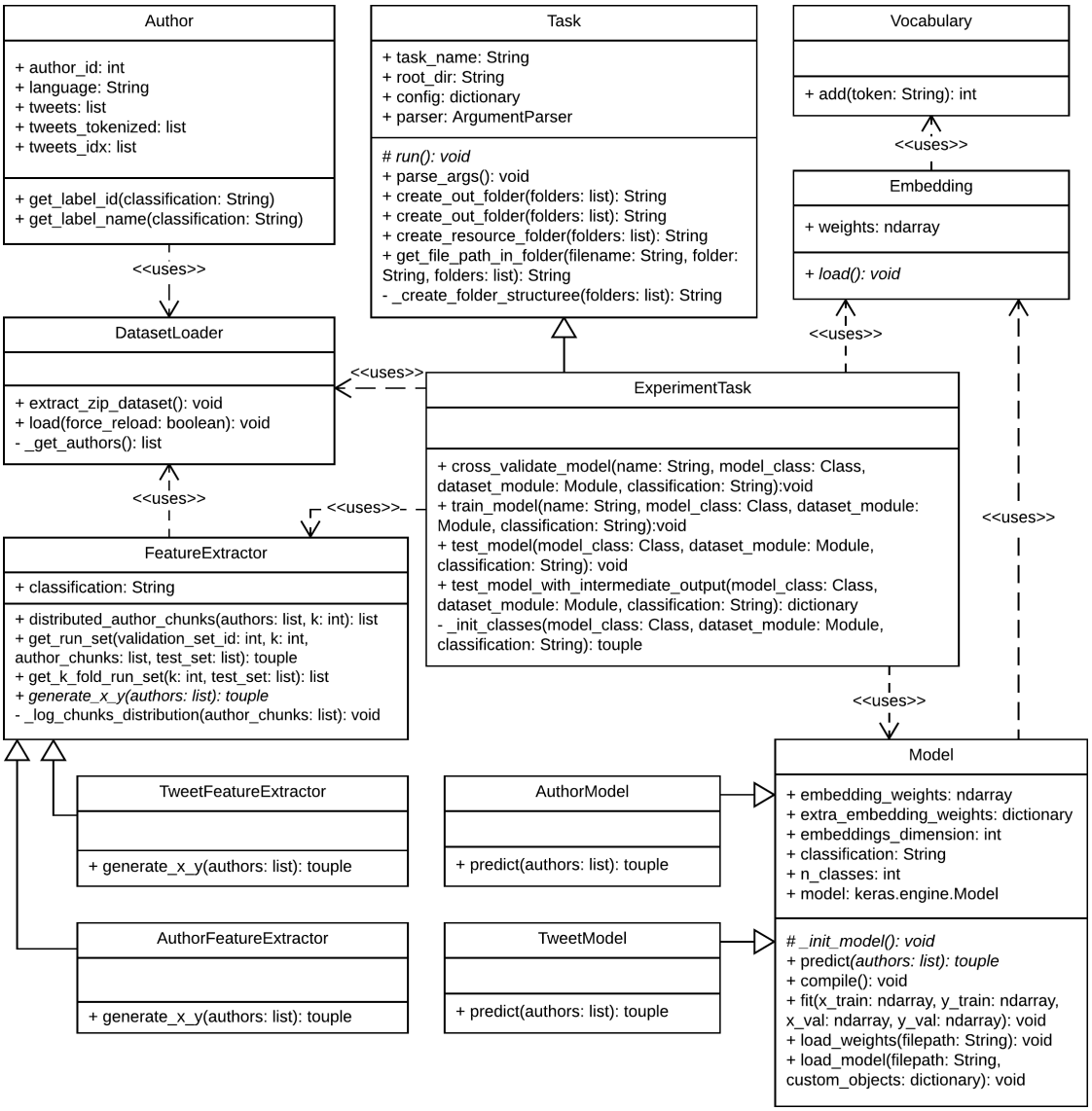

Abbildung 28: **Klassendiagramm der wichtigsten Komponenten der Projektstruktur.**

FeatureExtractor. Die `FeatureExtractor`-Klasse wird hauptsächlich von der `ExperimentTask`-Klasse angewendet, um aus den Datensätzen Features zu erstellen. In dieser Arbeit wird zwischen `TweetFeatureExtractor`- und `AuthorFeatureExtractor`-Klassen unterschieden.

Anwendung. Um diese Applikation zu benutzen muss Docker installiert sein. Um ein Modell auf einer Grafikkarte zu trainieren wird Nvidia-Docker empfohlen. Zuerst muss ein Container erzeugt werden mit dem folgenden Befehl:

```
docker build -f Dockerfile -t CONTAINER_NAME ..
```

Ein Task mit dem Namen `TASK_NAME` kann mit dem folgenden Befehl gestartet werden:

```
docker run -d -v %cd%:/data CONTAINER_NAME python /data/app/main.py TASK_NAME.
```

Falls Nvidia-Docker eingesetzt wird, muss `docker` mit `nvidia-docker` ersetzt werden.

Ordnerstruktur. In der Tabelle 27 wird die Ordnerstruktur beschrieben.

Ordner	Beschreibung
app	Dieser Ordner enthält die gesamte Python-Applikation.
notebooks	Dieser Ordner enthält Jupyter-Notebooks. Jupyter-Notebook ist nützlich, um Ausgaben zwischen den Schichten eines Modells interaktiv zu betrachten.
out	In diesem Ordner werden die Ausgaben und Resultate gespeichert.
resources	Dieser Ordner enthält alle Ressourcen, welche die Modelle benötigen.
resources/datasets	Dieser Ordner enthält verschiedene Datensätze. Sie können entweder als Zipdatei oder Ordner hinterlegt werden.
resources/embeddings	In diesem Ordner sind die Word Embeddings als Numpy-Dump und deren Vocabulary-Pickle gespeichert werden.
resources/models	In diesem Ordner sind die Modelle für ihre Verwendung zu finden.

Tabelle 27: **Beschreibung der Ordnerstruktur.**

Künftige Arbeit. Oft werden in Forschungsarbeiten wenig Gewicht auf eine saubere Projektstruktur gelegt. Wir sind jedoch der Meinung, dass durch eine gute Struktur viele Fehler vermieden und längerfristig viel Zeit eingespart werden kann. Aus diesen Gründen wurde in dieser Arbeit mehr Zeit für die Projektstruktur investiert, jedoch gibt es viele Möglichkeiten diese Struktur weiter zu verbessern. Einige Klassen haben noch eine starke Kopplung und das Evaluieren von mehreren Modellen in einem Durchlauf wird nicht unterstützt. Des Weiteren könnten neue Funktionalitäten von TensorFlow und andere Funktionalitäten, die den Workflow verbessern, implementiert werden.

9.3 Datenträger

Dies ist eine Beschreibung der Struktur sowie des Inhalts des dieser Arbeit zugehörigen Datenträgers.

- Bericht
  - Finale PDF-Version
- Dokumente
  - Paper (alle in dieser Arbeit zitierten Paper)
- System
  - author_profiling (Das in dieser Arbeit entwickelte System)
- Daten
  - PAN-2016
  - PAN-2017
  - Hardiyan-2017
  - PAN16b
  - Hardiyan17b
  - PAN16b+Hardiyan17b
  - PAN16b+Hardiyan17b:Limited
  - PAN16+Hardiyan17:Limited

10 Verzeichnisse

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[1] D. Kodiyan und F. Hardegger, *Weiss mein Computer wie alt ich bin? [Machine Learning]*, [Online; Zugriff am 5.6.2017]. Adresse: http://dreamboxx.com/mark/data/PABAs/PA16_Age_Gender_Detection_Kodiyan_Hardegger.pdf.

[2] J. Deriu und M. Cieliebak, „Sentiment Analysis using Convolutional Neural Networks with Multi-Task Training and Distant Supervision on Italian Tweets“, in *Evaluation of NLP and Speech Tools for Italian (EVALITA)*, 2016.

[3] L. Kong, C. Alberti, D. Andor, I. Bogatyy und D. Weiss, „DRAGNN: A Transition-based Framework for Dynamically Connected Neural Networks“, *CoRR*, Bd. abs/1703.04474, 2017.

[4] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola und E. Hovy, „Hierarchical Attention Networks for Document Classification“, in *Proceedings of NAACL-HLT*, 2016, S. 1480–1489.

[5] D. Bahdanau, K. Cho und Y. Bengio, „Neural Machine Translation by Jointly Learning to Align and Translate“, *CoRR*, Bd. abs/1409.0473, 2014.

[6] K. Cho, A. C. Courville und Y. Bengio, „Describing Multimedia Content using Attention-based Encoder-Decoder Networks“, *CoRR*, Bd. abs/1507.01053, 2015.

[7] S. Argamon, M. Koppel, J. Fine und A. R. Shimoni, „Gender, genre, and writing style in formal written texts“, *TEXT-THE HAGUE THEN AMSTERDAM THEN BERLIN-*, Bd. 23, Nr. 3, S. 321–346, 2003.

[8] A. Bartle und J. Zheng, „Gender classification with deep learning“, *Text-Interdisciplinary Journal*, 2015.

[9] A. Sboev, T. Litvinova, I. Voronina, D. Gudovskikh und R. Rybka, „Deep Learning Network Models to Categorize Texts According to Author's Gender and to Identify Text Sentiment“, in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, Dez. 2016, S. 1101–1106. DOI: 10.1109/CSCI.2016.0210.

[10] F. M. R. Pardo, P. Rosso, B. Verhoeven, W. Daelemans, M. Potthast und B. Stein, „Overview of the 4th Author Profiling Task at PAN 2016: Cross-Genre Evaluations“, in *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016.*, 2016, S. 750–784.

[11] A. Moujahid, *A Practical Introduction to Deep Learning with Caffe and Python*, [Online; Zugriff am 5.6.2017]. Adresse: http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/.

[12] cs231n, *CS231n Understanding Convolutions*, [Online; Zugriff am 5.6.2017]. Adresse: http://colah.github.io/posts/2014-07-Understanding-Convolutions/.

[13] ——, *CS231n Convolutional Neural Networks for Visual Recognition*, [Online; Zugriff am 5.6.2017]. Adresse: http://cs231n.github.io/convolutional-networks/.

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

[15] J. Chung, C. Gulcehre, K. Cho und Y. Bengio, „Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling“, *CoRR*, Bd. abs/1412.3555, 2014.

[16] F. Shaikh, *Introduction to Gradient Descent Algorithm along its variants*, [Online; Zugriff am 5.6.2017]. Adresse: https://www.analyticsvidhya.com/blog/2017/03/introduction-to-gradient-descent-algorithm-along-its-variants/.

[17] M. D. Zeiler, „ADADELTA: An Adaptive Learning Rate Method“, *CoRR*, Bd. abs/1212.5701, 2012.

[18] D. P. Kingma und J. Ba, „Adam: A Method for Stochastic Optimization“, *CoRR*, Bd. abs/1412.6980, 2014.

[19] S. Raschka, *Machine Learning FAQ*, [Online; Zugriff am 5.6.2017]. Adresse: https://sebastianraschka.com/faq/docs/evaluate-a-model.html.

[20] P. Bojanowski, E. Grave, A. Joulin und T. Mikolov, „Enriching Word Vectors with Subword Information“, *arXiv preprint arXiv:1607.04606*, 2016.

[21] T. Mikolov, I. Sutskever, K. Chen, G. Corrado und J. Dean, „Distributed Representations of Words and Phrases and their Compositionality“, *CoRR*, Bd. abs/1310.4546, 2013.

[22] J. Deriu, A. Lucchi, V. D. Luca, A. Severyn, S. Müller, M. Cieliebak, T. Hofmann und M. Jaggi, „Leveraging Large Amounts of Weakly Supervised Data for Multi-Language Sentiment Classification“, *CoRR*, Bd. abs/1703.02504, 2017.

[23] S. Mohammad und P. D. Turney, „Crowdsourcing a Word-Emotion Association Lexicon“, *CoRR*, Bd. abs/1308.6297, 2013.

[24] S. M. Mohammad und P. D. Turney, „Emotions Evoked by Common Words and Phrases: Using Mechanical Turk to Create an Emotion Lexicon“, in *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, Ser. CAAGET '10, Los Angeles, California: Association for Computational Linguistics, 2010, S. 26–34.

[25] F. Rangel und P. Rosso, „On the impact of emotions on author profiling“, *Information Processing & Management*, Bd. 52, Nr. 1, S. 73–92, 2016, ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2015.06.003.

[26] W. Zeng, W. Luo, S. Fidler und R. Urtasun, „Efficient Summarization with Read-Again and Copy Mechanism“, *CoRR*, Bd. abs/1611.03382, 2016.

[27] A. Radford, R. Józefowicz und I. Sutskever, „Learning to Generate Reviews and Discovering Sentiment“, *CoRR*, Bd. abs/1704.01444, 2017.

[28] J. P. C. Chiu und E. Nichols, „Named Entity Recognition with Bidirectional LSTM-CNNs“, *CoRR*, Bd. abs/1511.08308, 2015.

[29] Martı n Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu und Xiaoqiang Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, Software erhältlich auf tensorflow.org, 2015. Adresse: http://tensorflow.org/.

Zürcher Hochschule für Angewandte Wissenschaften School of Engineering

[ONLINE ADMINISTRATION PRAKTISCHE ARBEITEN]

[DEPT. T ADMIN TOOLS]

zurück Logout

Bachelorarbeit 2017 - FS: BA17_ciel_6

Allgemeines:

Titel: Age und Gender Detection mit Word Embeddings und Deep Learning
Anzahl Studierende: 2

Betreuer:

HauptbetreuerIn: Mark Cieliebak, ciel
NebenbetreuerIn: Stephan Neuhaus, neut

Zugeteilte Studenten:

Diese Arbeit ist zugeteilt an:
- Florin Hardegger, hardeflo (IT)
- Don Anson Kodiyan, kodiydon (IT)

Fachgebiet:

DA Datenanalyse
SOW Software

Studiengänge:

IT Informatik

Zuordnung der Arbeit :

InIT Institut für angewandte Informationstechnologie

Infrastruktur:

Arbeitsplatz wird durch InIT zur Verfügung gestellt

Interne Partner :

Es wurde kein interner Partner definiert!

Industriepartner:

Es wurden keine Industriepartner definiert!

Beschreibung:

Es klingt fast nach Zauberei: man gibt dem Computer ein paar Zeilen Text einer Person, und schon erkennt er das Alter und Geschlecht des Autors (bzw. der Autorin)! Wenn dies zuverlässig funktioniert, kann man z.B. Fake-Accounts auf Facebook erkennen und ausschliessen.

Aber wie macht man das? Wir verwenden einen Ansatz mittels Deep Learning, den wir gemeinsam mit der ETH entwickelt haben: Zunächst erzeugt der Computer aus Milliarden Beispiel-Texten ein generisches Sprachmodell. Dieses wird anschliessend für Alters- bzw. Geschlechtserkennung trainiert und optimiert wird.

Einen analogen Ansatz haben wir bereits für die Sentiment-Analyse (ist ein Text positiv oder negativ) eingesetzt und damit SemEval 2016 gewonnen. Auch für die Alterbestimmung gibt es einen Wettbewerb, PAN, an dem wir dieses Jahr zum ersten Mal teilgenommen haben (Resultate ausstehend). Unser Ziel ist es den nächsten Wettbewerb, PAN 2017, zu gewinnen!

In dieser Arbeit adaptieren Sie die existierenden Technologien für die Alters/Geschlechtsbestimmung. Dazu verwenden Sie Standard-Libraries für Deep Learning (zB Theano), unsere IT Infrastruktur mit GPU-Maschinen, einen Datenpool von über einer Milliarde Tweets sowie verschiedenen Methoden aus dem Machine Learning.

zurück Logout