FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Automatische Bewertung von Textkomplexität

Patrick Gustav Blaneck, 3283112

Erstbetreuer: Prof. Dr. rer. nat. Stephan Bialonski

Zweitbetreuer: Niklas Grieger, M. Sc.

MATSE
Ausbildung

5. September 2022

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die Seminararbeit mit dem Thema

Automatische Bewertung von Textkomplexität

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Ich verpflichte mich, ein Exemplar der Seminararbeit fünf Jahre aufzubewahren und auf Verlangen dem Prüfungsamt des Fachbereiches Medizintechnik und Technomathematik auszuhändigen.

Name: Patrick Gustav Blaneck

Aachen, den 5. September 2022

_____

Unterschrift des Studierenden

Zusammenfassung

Zuverlässiges Vorhersagen von Textlesbarkeit kann weitreichenden Einfluss auf verschiedenste Bereiche haben, so z. B. auf maschinelle Übersetzung oder auch selbstüberwachtes Lernen. Vor kurzem wurden deutsche LLMs verfügbar, die Deep Learning-basierte Methoden zur automatischen Lesbarkeitsanalyse möglich machen und vor allem die bisher genutzten Ansätze übertreffen. Besonders nennenswert sind hier Modelle wie GBERT und GPT-2-Wechsel.

Diese Seminararbeit fokussiert sich auf die Fähigkeit von vortrainierten GBERT- und GPT-2-Wechsel-Ensembles, zuverlässig die Komplexität von deutschen Sätzen vorherzusagen. Die genannten Sprachmodelle wurden mit diversen linguistischen Features kombiniert und auf einen Zusammenhang zwischen Vorhersageleistung und Ensemblegröße bzw. -zusammensetzung untersucht.

Ensembles aus einem Verbund aus GBERT- und GPT-2-Wechsel-Modellen schnitten dabei besser ab, als homogene Ensembles des gleichen Umfangs.

Diese Arbeit wurde im Rahmen der KONVENS 2022 durchgeführt und die genannten Ansätze auf dem *Text Complexity DE 2022*-Wettbewerbsdatensatz, bestehend aus deutschen Sätzen, ausgewertet und erreichte auf Testdaten einen RMSE (Root Mean Squared Error) von 0.435.

Inhaltsverzeichnis

1 Einleitung

Automatische Lesbarkeitsanalyse (engl. *Automatic Readability Assessment (ARA)*) ist eine wohlbekannte Herausforderung in der Natural Language Processing-Forschung [16, 32, 7]. Die präzise Vorhersage von Textlesbarkeit hat das Potential lerneingeschränkte Lesende [25] und selbstüberwachtes Lernen zu unterstützen [2], oder die Komplexität von maschinengenerierten Übersetzungen zu regulieren [1].

Die bisherigen Entwicklungen der Lesbarkeitsanalyse lässt sich in drei Phasen teilen:

1. Klassische Lesbarkeitsformeln basieren auf statistischen Messwerten linguistischer Semantik (insbesondere lexikalische bzw. Satzsemantik) und syntaktischen Merkmalen. Fortschritte im Natural Language Processing konnten diese Ergebnisse dann mithilfe linguistischer Kohärenz (logischer Zusammenhang von Sätzen) oder komplexeren Merkmalen aus der lexikalischen Semantik (Wortbedeutung) [16] übertreffen.
2. Im frühren 21. Jahrhundert wurden selbsterstellte linguistische Merkmale für das Training „flacher“[1] Klassifikatoren und Regressoren, wie Support Vector Machines oder Entscheidungsbäume, genutzt, die die Präzision noch einmal verbesserten [7].
3. Die aktuellste Entwicklungsphase ist geprägt von *Large Language Models (LLMs)* aus der Deep Learning-Sphäre. Diese Sprachmodelle erlernen selbstständig Textmerkmale in Form von Merkmalsvektoren durch das Einspeisen von großen Textkorpora während selbstüberwachtem Vortraining (engl. *self-supervised pretraining*). Eines der einflussreichsten Modelle dieser Art stellt Konzept des Transformers [33] dar, das effizient linguistische Kohärenz auch von linear weit entfernten[2] Textkomponenten erkennen lässt. Erfolgreiche Implementierungen dieses Modells für die englische Sprache sind etwa BERT [8, 27] oder GPT [23, 24, 4]. Die Kombination dieses Ansatzes mit dem der selbsterstellten linguistischen Merkmale führte dann zu noch einmal verbesserten Leistungen zur Lesbarkeitsanalyse englischer Texte [14, 13].

Voraussetzung für das erfolgreiche Training dieser LLMs ist die Existenz umfassender Textkorpora der Zielsprache, was ein Hindernis in weniger verbreiteten Sprachen[3] darstellt. Aus diesem Grund basieren bisher die meisten Ansätze zur automatischen Lesbarkeitsanalyse im Deutschen auf selbsterstellen linguistischen Merkmalen und klassischen Methoden des maschinellen Lernens, wie polynomieller Regression, Support Vector Machines oder Random Forests [11, 35, 21, 34].

Nichtsdestotrotz existieren mittlerweile LLMs auch für die deutsche Sprache, wie

[1] Explizite Abgrenzung zum *Deep Learning*.

[2] Lineare Entfernung beschreibt die bloße syntaktische Distanz zwischen Textkomponenten, meist innerhalb eines Satzes. Im Kontrast dazu steht die hierarchische Entfernung, die die Distanz im Syntaxbaum beschreibt [37].

[3] Im Vergleich zu Englisch, so etwa Deutsch.

GBERT [5] (basierend auf BERT) und GPT-2-Wechsel [19] (basierend auf dem englischen GPT-2-Modell [24]). Bisher ist noch nicht bekannt, zu welchem Grade diese deutschen LLMs die Fähigkeit besitzen, die Lesbarkeit von deutschen Texten vorherzusagen.

Diese Seminararbeit befasst sich mit der Frage, inwieweit Ensembles bestehend aus GBERT- und GPT-2-Wechsel-Modellen die Fähigkeit besitzen, die Lesbarkeit von deutschen Texten vorherzusagen. Diese Ensembles werden mit klassischen linguistischen Merkmalen kombiniert und der daraus entstandene Ansatz wird auf einem kürzlich publizierten Datensatz deutscher Sätze evaluiert [20].

Analog zu bisheriger Arbeit im Gebiet der LLM-Ensembles [26, 3] wird untersucht, ob Vorhersageleistung und Ensemblegröße und -zusammensetzung zusammenhängen.

Die Arbeit wurde im Rahmen des *Text Complexity DE Challenge 2022*-Shared Tasks der KONVENS 2022[4] durchgeführt. Die Implementierung der durchgeführten Experimente ist online verfügbar[5].

2 Datengrundlage und Aufgabe

> Bei der Tour de France liegt die höchste Durchschnittsgeschwindigkeit eines Fahrers bei 41 km/h. (MOS: 1.5)
>
> Für die Union resultiert daraus sowohl ein Akzeptanzproblem bei den EU-Bürgern, denen „Brüssel“ immer undurchsichtiger erscheint, als auch die mit dem Mitgliederwachstum verbundene Schwierigkeit, im bestehenden Institutionengefüge die Arbeits- und Handlungsfähigkeit der einzelnen Organe zu gewährleisten. (MOS: 6.33)

Abbildung 1: Beispiele aus dem Datensatz des *Text Complexity DE Challenge 2022*-Shared Tasks. Werte in Klammern beschreiben Bewertungen der Satzkomplexität.

Der *Text Complexity DE 2022*-Wettbewerbsdatensatz besteht aus 1000 gelabelten Sätzen und war gegebener Bestandteil der KONVENS 2022 [20]. Die Sätze wurden aus 23 Artikeln der deutschen Wikipedia gewonnen. 250 dieser Sätze wurden dann manuell von Deutsch-Muttersprachler:innen vereinfacht [20].

Die Bewertungen (Label) der Sätze wurden durch ein Online-Umfragetool erhalten; dabei wurden Teilnehmende gebeten, die allgemeine Komplexität, Verständlichkeit und die lexikalische Schwierigkeit (Komplexität des Vokabulars) auf einer Likert-Skala von 1 (geringster Wert) bis 7 (höchster Wert) zu bewerten [20]. Insgesamt resultierten daraus 10650 Satzbewertungen verteilt über die insgesamt 1000 Sätze.

[4] Konferenz zur Verarbeitung natürlicher Sprache, siehe https://konvens2022.uni-potsdam.de/.
[5] Siehe https://github.com/dslaborg/tcc2022.

Nach einer Datenbereinigung wurden fünf bis 18 Bewertungen akzeptiert und folgend genutzt um einen Durchschnittswert, den *Mean Opinion Score (MOS)*, pro Metrik zu erhalten [20].

Die Aufgabe bestand daraus, den MOS der Komplexität eines Satzes zu bestimmen. Der MOS ist ein arithmetisches Mittel (siehe Abbildung 1), daher handelt es sich hier um einen Dezimalwert und damit generell um eine Regressionsaufgabe.

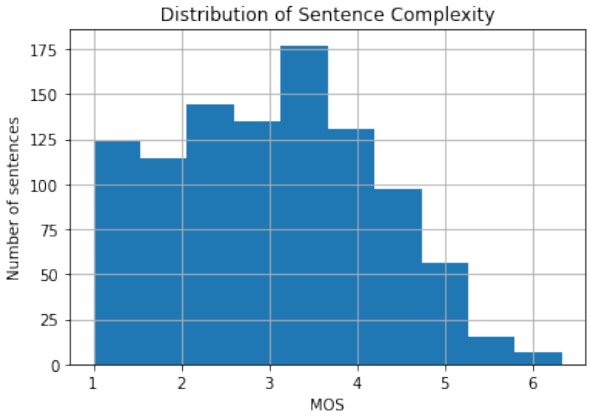

Abbildung 2: Histogramm des *Mean Opinion Scores (MOS)* der Satzkomplexität für den gegeben Datensatz.

Die Verteilung der Komplexitätsbewertungen vermittelt dabei, dass komplexere Sätze im Datensatz deutlich seltener vertreten sind als simplere (siehe Abbildung 2).

3 Theoretische Grundlagen

3.1 Rekurrente neuronale Netze und Backpropagation

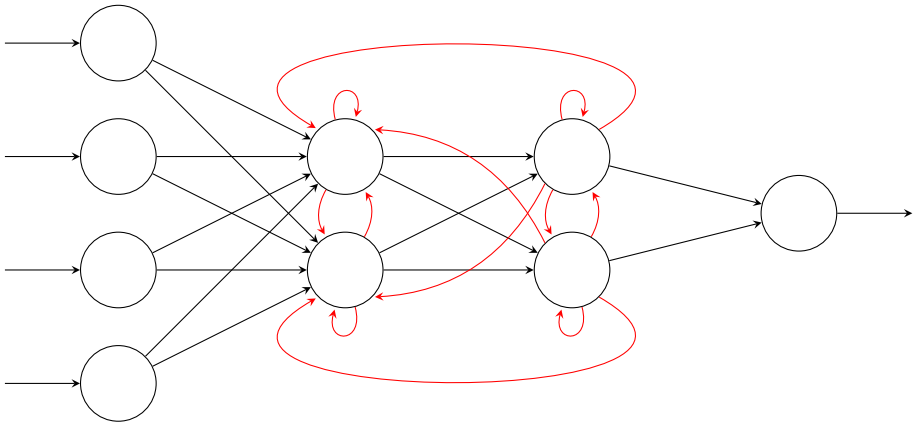

Abbildung 3: Darstellung eines einfachen rekurrenten neuronalen Netzes. Rekurrente Kanten sind rot gekennzeichnet.

Im Gegensatz zu klassischen künstlichen neuronalen Netzen bei denen Neuronenausgaben ausschließlich in Verarbeitungsrichtung weitergegeben werden (*feed-forward*-Netze), werden bei rekurrenten neuronalen Netzen auch rückgerichtete (*rekurrente*) Kanten zugelassen und erlauben somit Feedback Loops [28] (siehe Abbildung 3).

Diese Rückverbindungen sorgen damit dafür, dass bisherige Ergebnisse die folgenden beeinflussen können und statten das Netz mit einem „Gedächtnis“ aus. Insbesondere sind rekurrente neurale Netze damit ein ideales Werkzeug für das Verarbeiten von zusammenhängenden Sequenzen, wie z. B. beim Verarbeiten natürlicher Sprache.

Das Training der Gewichte von feed-forward-Netzen erfolgt zumeist über das Verfahren der *Backpropagation* (siehe Abbildung 4), ein Spezialfall des *Gradient Descent*. Backpropagation berechnet dabei den Gradienten der Fehlerfunktion mithilfe der Kettenregel rückwärts-iterierend Layer für Layer.

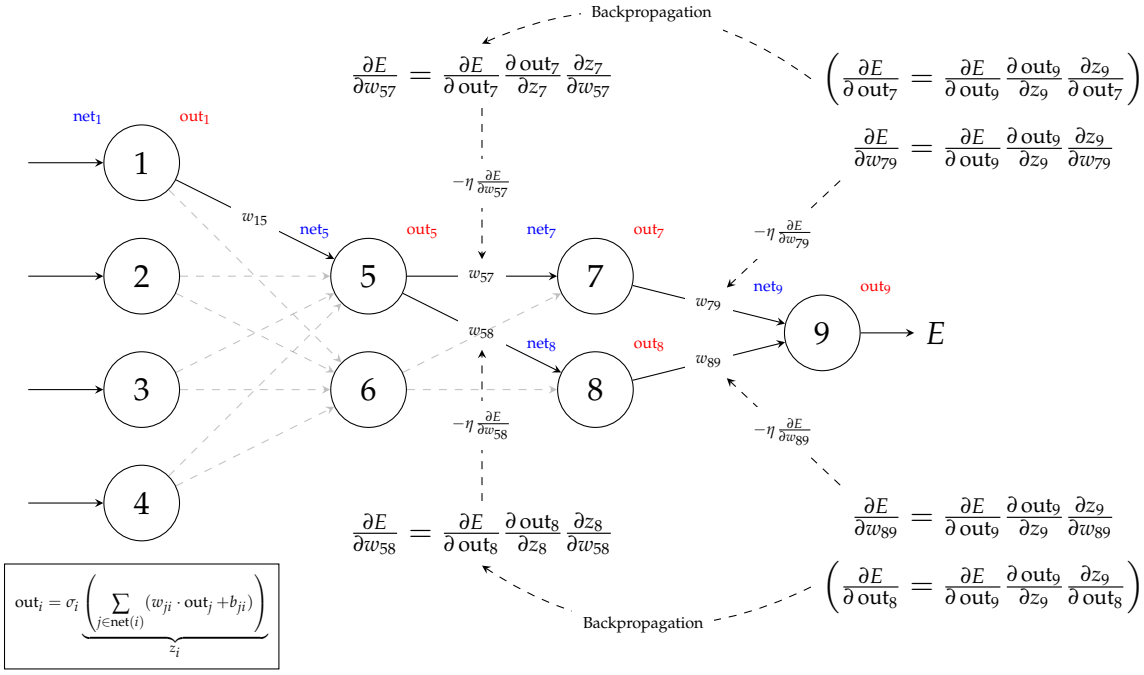

Abbildung 4: Darstellung der Backpropagation für einige Gewichte w_{ji} in einem einfachen feed-forward-Netz. Der Algorithmus erfolgt analog für den Bias. Für jedes Neuron i stellt net_i den Input der jeweiligen Input-Neuronen und out_i den Output des Neurons bzw. den Wert der Aktivierungsfunktion σ_i dar. η gibt die Schrittweite an. Das Neuron *9* liefert die Gesamtvorhersage des Netzes und mit dem realen Label y berechnet sich damit der Fehler $E = (\text{out}_9 - y)^2$. Man erkennt, dass durch die Backpropagation und dabei insbesondere die Kettenregel pro Gewicht nur kleine zusätzliche Berechnungen nötig sind, um dieses anzupassen. Die Berechnungen sind dabei sehr gut parallelisierbar und damit sehr effizient.

Ein Spezialfall der Backpropagation für rekurrente neuronale Netze ist *Backpropagation Through Time (BPTT)*. Beim Training mithilfe von BPTT werden rekurrente Verbindungen „entfaltet“, wodurch das RNN durch ein feed-forward-Netz approxi-

miert wird. Die Anzahl der entfalteten bzw. erzeugten Neuronen ist dabei abhängig von der Länge der Trainingsdaten (siehe Abbildung 5). Jede Entfaltung stellt eine Zeiteinheit dar. Das Verfahren ist dann analog zur regulären Backpropagation, wobei die entfalteten Neuronen und deren Gewichte entsprechend angepasst werden.

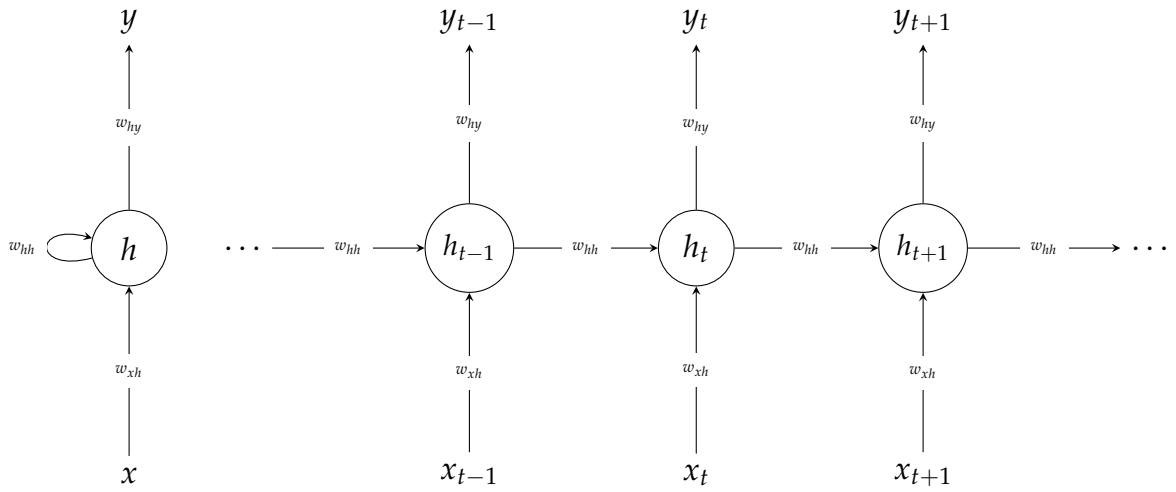

Abbildung 5: Darstellung eines einfachen entfalteten RNNs um einen Zeitpunkt t.

Ein großes Problem stellt hierbei das des *Vanishing* bzw. *Exploding Gradients* dar, das insbesondere bei langen Eingabesequenzen auftritt, da proportional zur Länge der Eingabesequenz entsprechend viele Ableitungen der jeweiligen Aktivierungsfunktion gebildet werden müssen. Zusätzlich ist eine komplette Backpropagation auch sehr rechenintensiv.

Eine Lösung ist *Truncated Backpropagation Through Time (TBPTT)*. Dabei wird der BPTT-Algorithmus so modifiziert, dass nach k_1 Vorwärtsiterationen auf k_2 Zeiteinheiten Backpropagation Through Time angewandt wird. Dabei sollte k_2 insbesondere so groß gewählt werden, dass alle relevanten Zusammenhänge abgebildet werden können[6]. Es entstehen folgende Varianten TBPTT(k_1, k_2):

- TBPTT(n, n): Klassische Backpropagation Through Time.
- TBPTT$(1, n)$: Nach jeder Zeiteinheit wird die Backpropagation auf alle bisherigen Zeitpunkte angewandt [36].
- TBPTT(k_1, k_2) $(k_2 < k_1 < n)$: Mehrere Updates pro Sequenz, gleich große Zeitfenster resultieren in stabileren Ergebnissen [18].
- TBPTT(k_1, k_2) $(k_1 < k_2 < n)$: Es werden pro Sequenz mehrere Updates durchgeführt, jedes Neuron wird mehr als einmal geupdated [36].
- TBPTT(k_1, k_2) $(k_1 = k_2)$: Gebräuchlichster Fall; Kompromiss aus Stabilität und Schnelligkeit. [10, 29, 22]

[6] Bei $k_2 = 20$ würden temporale Zusammenhänge nach 20 Zeiteinheiten nicht mehr erkannt werden. [31]

RNNs besitzen durch die rekurrenten Kanten die womöglich einfachste Art und Weise, Outputs von Neuronen wiederzuverwerten – die bereits erwähnten Feedback-Loops. Durch das Problem des Vanishing Gradients ist dieses Gedächtnis allerdings nur sehr kurzlebig und es können nur schlecht Schlüsse über kontextuellen Zusammenhänge über längere Distanzen in einer Eingabesequenz getroffen werden. Eine Erweiterung stellen *Long Short Term Memory Networks (LSTMs)* [12] dar, die dieses Gedächtnis verbessern; auf diese soll hier aber nicht weiter eingegangen werden. Das Training von RNNs (und auch LSTMs) ist sehr aufwändig und nicht parallelisierbar. Parallelisierbare Architekturen sind insbesondere bei großen verfügbaren Datensätzen und großen Modellen vorzuziehen.

3.2 Attention und Transformer

Insbesondere in der natürlichen Sprache ist es wichtig zu bestimmen, wie einzelne Komponenten sich kontextuell zueinander verhalten. Im Machine Learning existiert dafür das Konzept der *Attention*. Dabei wird z. B. in einem *tokenisierten* Eingabesatz für jedes Token bestimmt, wie relevant jedes andere im Kontext ist.

Eine Attention-Funktion benötigt in der hier betrachteten Variante drei Inputs: *Query*, *Key* und *Value* [33]. Die Aufgabe der Funktion besteht darin, die relevanten[7] Token aus den Keys zu einer Query zu bestimmen. Die „Ähnlichkeit" zweier Vektoren lässt sich bekanntermaßen einfach mithilfe des Skalarproduktes bestimmen. Um numerische Ungenauigkeiten zu vermeiden, werden die Vektoren oft mit $\sqrt{d}$ skaliert, wobei d die Dimension des jeweiligen Merkmalsraum angibt. Diese Form der Attention heißt *Scaled Dot-Product Attention* [33].

Ein Spezialfall der Attention tritt auf, wenn Queries, Keys und Values aus der gleichen Domäne stammen. Man spricht von *Self-Attention* [33].

Das Skalarprodukt wird in eine Softmax-Funktion gegeben und das Ergebnis stellt den *Attention Score* dar [33]. Der Attention Score dient als Gewichtung der Value-Vektoren und durch entsprechende Linearkombination werden schließlich kontextualisierte Embeddings erzeugt. Es gilt also [33]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

Statt einer einzelnen Attention-Operation über den gesamten Merkmalsraum werden häufig mehrere Operationen in Form der *Multi-Head Attention* durchgeführt. Bei Multi-Head Attention werden pro Attention-Head i die ursprünglichen Embeddings mithilfe trainierbarer Gewichte W_i^Q, W_i^K, W_i^V in oft kleinere Räume für Queries, Keys und Values projiziert [33]. Diese Projektion stellt dabei eine Restriktion des Kontextes dar; so können mehrere Attention-Heads auch mehrere kontextuelle Zusammenhänge

[7] Relevant bedeutet hier „räumlich nah im Merkmalsraum" und damit auch bedeutungsnah.

finden, was bei einer allgemeinen und damit oberflächlicheren Attention-Operation nicht möglich ist. Verschiedene Attention-Heads können z. B. Objekte mit den zugehörigen Verben oder Präpositionen verbinden, Substantive mit den zugehörigen Attributen oder stellen Verbindungen zu Satzzeichen her [6]. Die Ergebnisse der einzelnen Attention-Heads werden dann konkateniert und noch einmal projiziert mit Gewichten W^O. Es gilt also [33]:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$

$$\text{mit head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

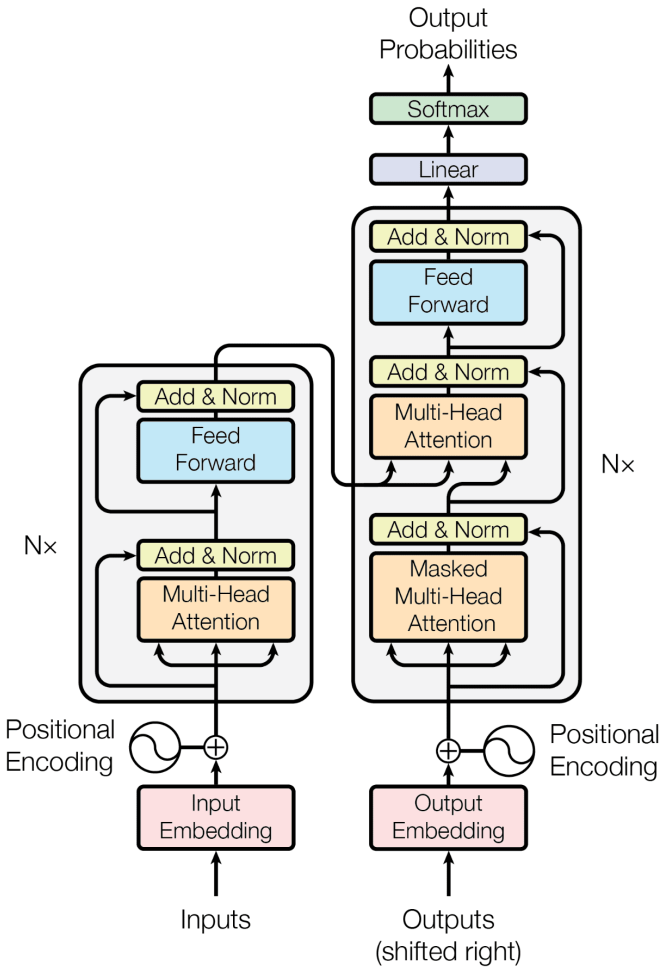

Abbildung 6: Darstellung des Transformer-Modells aus [33].

Eine Anwendung der Attention ist der *Transformer* [33]. Der Transformer dient in erster Linie dazu, eine Input-Sequenz in eine Output-Sequenz zu übersetzen, wobei die Operationen sehr gut parallelisierbar sind. Er besteht aus einer Encoding- und einer Decoding-Einheit, wobei beide aus mehreren nacheinander geschalteten Encodern

bzw. Decodern besteht. Diese bestehen widerum aus (Self-)Attention-Units und Feed-Forward-Netzen (siehe Abbildung 6).

Jede Input-Sequenz wird zuerst tokenisiert und mit einem Positional Encoding versehen[8]. Diese Embeddings mit Positional Encoding werden dem ersten Encoder als Input gegeben. Die Aufgabe der Encoder-Einheit besteht daraus, insbesondere mithilfe von Self-Attention, aus dem Input Merkmale zu extrahieren, die dann vom Decoder verwendet werden, um einen Output zu generieren. [33]

Dabei erhält jeder Decoder den Output des letzten Encoders als Input, sowie den Output des vorherigen Decoders (siehe Abbildung 6). Dem Self-Attention-Layer in einem Decoder ist es dabei allerdings nur erlaubt, auf vorherige Positionen der Ausgabesequenz zu reagieren. Dazu werden entsprechend die zukünftigen Positionen vor der Softmax-Funktion mit $-\infty$ aufgefüllt. [33]

4 Vorgehen

4.1 Vorbereitung und Datenaufteilung

Der gesamte Datensatz bestehend aus Trainings-, Validierungs- und Testdatensatz wurden identisch behandelt. Je nach benutztem Modell wurden die Sätze mit dem modellspezifischen Tokenizer behandelt, wodurch jedes Token auf ein spezifisches Embedding zeigt. Dabei wurden alle Sätze auf eine uniforme Länge von 128 Token gepadded[9].

Während der Explorationsphase wurden die Modelle jeweils mit einer fünffachen Kreuzvalidierung evaluiert, wobei jeder Teildatensatz 20 % der zufällig sortierten Trainingsdaten enthielt. In jeder Teilmenge wurden weiterhin 10 % der Daten, also 8 % der gesamten Trainingsdaten, als Early-Stopping-Set gewählt. Insgesamt wurden die Modelle in dieser Phase also auf jeweils 72 % der Trainingsdaten trainiert.

Um die Ergebnisse noch einmal zu verbessern, wurden die vielversprechendsten Modelle zur Einreichung für die *Text Complexity Challenge DE 2022* auf dem gesamten Trainingsdatensatz trainiert. Dabei wurden erneut 7.5 % als Early-Stopping-Set verwendet. Damit wurden die finalen Modelle auf 92.5 % der Trainingsdaten trainiert.

4.2 Lesbarkeitsmerkmale

Zusätzlich zu den Outputs der Modelle wurden traditionelle Lesbarkeitsmerkmale ausgewertet. Um diese Merkmale zu erzeugen, wurden die beiden Projekte

[8] Dieses *Positional Encoding* sorgt dafür, dass das gleiche Token an verschiedenen Stellen in einer Sequenz anders behandelt wird.

[9] *Padding* beschreibt hier, dass Sätze mit weniger als 128 Token mit entsprechenden Padding-Tokens versehen werden. So haben alle Inputs des Modells die gleiche Länge.

Modell	`gbert-large`	`gpt-2-xl-wechsel-german`
Aufgaben	Language Modeling Next Sentence Prediction	Textgenerierung
Vokabular (Token)	31.000	50.000
Parameter	336 Millionen	1,5 Milliarden
Hidden State Size	1024	1600
Größe	$\approx$ 1.35 GB	$\approx$ 6.28 GB

Abbildung 7: Vergleich einiger Kennzahlen der Modelle `gbert-large` und `gpt-2-xl-wechsel-german`.

`andreasvc/readability`[10] und `tsproisl/textcomplexity`[11] verwendet. Diese bestimmen unter anderem bereits definierte Lesbarkeitsmetriken, auf die hier aber nicht weiter eingegangen werden soll, wie z. B.

- den *Automated Readability Index* [30], eine Formel zum Berechnen der nötigen Schulstufe um einen Satz zu verstehen,
- die ähnliche *Simple Measure of Gobbledygook (SMOG)* [17], oder
- den *Flesch Reading-Ease Score (FRES)*, der eine generelle Komplexitätsbewertung bestimmt. [9]

Es werden aber auch oberflächlichere Eigenschaften, wie die Satzlänge, Zeichensetzung, oder die Höhe des Syntaxbaumes eines Satzes bestimmt.

Um noch mehr Merkmale zu berechnen, wurden diese Merkmale einmal für die gegebenen deutschen Sätze und deren englische Übersetzungen berechnet. Es wurde also davon ausgegangen, dass die Sätze auch im Englischen eine ähnliche Komplexität aufweisen.

Über beide Sprachen hinweg wurden insgesamt 154 zusätzliche Features ermittelt.

4.3 Modelle

Untersucht wurden die beiden deutschen Sprachmodelle GBERT [5] (basierend auf BERT [8]) und GPT-2-Wechsel [19] (basierend auf GPT-2 [23]). Genutzt wurden die Gewichte der bereits vortrainierten Modelle `gbert-large`[12] und `gpt-2-xl-wechsel-german`[13] (siehe Abbildung 7).

Für das GBERT-Modell wurde allen Sätzen ein Klassifizierungstoken (`[CLS]`) vorangestellt, welches auch im Vortraining für die Aufgabe der *Next Sentence Prediction* verwendet wurde [8].

[10] Siehe https://github.com/andreasvc/readability.
[11] Siehe https://github.com/tsproisl/textcomplexity.
[12] Siehe https://huggingface.co/deepset/gbert-large.
[13] Siehe https://huggingface.co/malteos/gpt2-xl-wechsel-german.

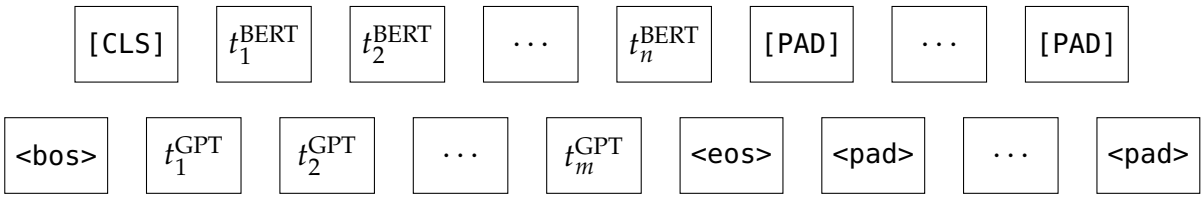

Abbildung 8: Tokenisierung von Sätzen für die Sprachmodelle GBERT (oben) und GPT-2-Wechsel (unten).

GPT-basierte Modelle sind in der Regel nicht für Regressionsaufgaben konzipiert, weshalb der Tokenizer wie folgt angepasst wurde:

- Einführung eines neuen Padding-Tokens (`<pad>`) zum padden auf uniforme Länge von 128,
- Voranstellung eines *Beginning of Sequence*-Tokens `<bos>`,
- Anfügen eines *End of Sequence*-Tokens `<eos>` an das Ende eines jeden tokenisierten Satzes.

Um die Modelle für die *Text Complexity Challenge DE 2022* zu optimieren, wurden folgende Multi-Layer Perzeptrons (MLPs) als Head auf den Modellen genutzt:

- Zum Finetunen wurde ein MLP bestehend aus einem linearen Layer eingesetzt, das alle Outputs des letzten (Hidden) Layers des Modells erhält und dann mithilfe des Mean Squared Errors den Fehler bestimmt[14].
- Um die erstellten Lesbarkeitsmerkmale ebenfalls zu nutzen, wurde nach dem Finetunen ein weiteres MLP trainiert, das die letzten (Hidden) Layer des jeweiligen Modells und zusätzlich die erstellten Lesbarkeitsmerkmale als Input erhält. Dieses MLP besteht aus einem linearen Layer gefolgt von einem ReLU[15]-Aktivierungs- und dem Output-Layer.

4.4 Training und Ensembling

Die Modelle werden im Kontext der *Text Complexity Challenge DE 2022* mit dem *Root Mean Squared Error (RMSE)* evaluiert. Der RMSE für N Datenpaare (x_i, y_i) ist gegeben mit

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

wobei y_i den tatsächlichen und $\hat{y}_i$ den bestimmten Wert für eine Eingabe x_i darstellt. Während der Explorationsphase wurde der RMSE für jeden Teildatensatz der fünffachen Kreuzvalidierung berechnet.

Das Training wurde in zwei Phasen durchgeführt:

[14] Siehe huggingface/transformers für BERT bzw. huggingface/transformers für GPT2.
[15] $\text{ReLU}(x) = \max\{0, x\}$

Die erste Phase bestand aus dem Finetuning des jeweiligen Modells mithilfe eines MLPs als Regression-Head. Genutzt wurde dabei ein AdamW-Optimizer[16] [15] mit einer Batch-Size von 16 und einer Lernrate von $\eta = 5 \cdot 10^{-5}$ mit linearem Warmup auf den ersten 30% der Trainingsschritte von 0 bis η. Nach einer halben Trainingsepoche wurde mithilfe des Early-Stopping-Sets der Fehler berechnet und mithilfe von Backpropagation die Gewichte im Modell und dem MLP angepasst. Dies wurde wiederholt, bis sich entweder der RMSE für fünf Evaluationen nicht verkleinert hat, oder über 100 Epochen trainiert wurde[17].

In der zweiten Phase wurden die Regression-Heads verworfen und der letzte Hidden State extrahiert. Für GBERT wurde der Output des Klassifizierungstokens benutzt, für GPT-2-Wechsel der des End of Sequence-Tokens. Der gesamte Merkmalsvektor bestand dann aus dem jeweiligen Modell-Output und den Lesbarkeitsmerkmales pro Satz. Das dafür genutzte MLP hatte zwei Layer (linear und ReLU-Aktivierung), nutzte den RMSprop-Optimizer, eine Batch-Size von 16 und eine Lernrate von $\eta = 10^{-3}$. Analog zu den Modellen wurde das MLP auf dem Early Stopping-Set nach jeder Epoche evaluiert. Erneut wird entweder nach 5000 trainierten Epochen oder 100 Epochen ohne Verbesserung abgebrochen um Overfitting zu vermeiden. Anschließend wurde das Modell mit dem geringsten RMSE auf dem Early Stopping-Set zurückgegeben.

Die genutzten Sprachmodelle sind sehr groß und der Datensatz dafür verhältnismäßig klein. Das führt zu einer großen Varianz und zu Overfitting in den Vorhersagen der einzelnen Modelle. Um diesem Effekt entgegenzuwirken, wurden Ensembles der Modelle gebildet [26, 3]. Jedes Mitglied eines Ensembles wurde mithilfe jeweils zufällig ausgewählten Trainings- und Early Stopping-Sets trainiert. Die Gesamtaussage eines Ensembles wird dann durch das arithmetische Mittel der Vorhersagen der einzelnen Mitglieder gebildet.

4.5 Nachbereitung

Während der Analyse wurde erkenntlich, dass einige Modelle Werte kleiner als 1.0 vorhersagen. Dies ist aber aufgrund der verwendeten Likert-Skala von 1 (geringster Wert) bis 7 (höchster Wert) und dem daraus gebildeten arithmetischen Mittel unmöglich. Diese Werte gingen beim Ensembling entsprechend nicht ein. Möglicherweise fand eine Verteilungsverschiebung bei den generierten Lesbarkeitsmerkmalen statt.

5 Ergebnis

Während der Explorationsphase wurde die Leistungsfähigkeit von Ensembles verschiedener Größe und Komposition evaluiert. Die Ensembles bestanden aus insgesamt

[16] Siehe huggingface/transformers.

[17] Um Underfitting zu vermeiden, wurde dieser Mechanismus für die ersten 300 Schritte (ca. zehn Epochen) nicht angewandt.

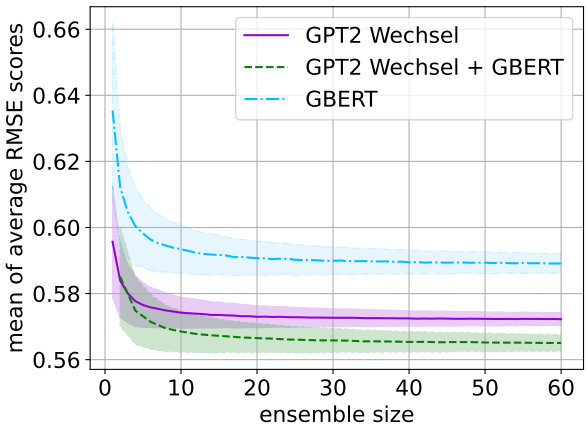

Abbildung 9: Darstellung des der Abhängigkeit des RMSE von der Ensemblegröße für verschiedene Ensemblekompositionen. Das arithmetische Mittel ist als Linie, die Standardabweichung als Schattierung dargestellt.

bis zu 60 Modellen, bestehend aus:

1. ausschließlich GBERT-Modellen,
2. ausschließlich GPT-2-Wechsel-Modellen, oder
3. gleicher Anzahl von GBERT- und GPT-2-Wechsel-Modellen.

Die Abhängigkeit zur Ensemblegröße wurde mithilfe von Bootstrapping[18] bestimmt [26, 3]. Insgesamt wurden jeweils 100 GBERT- und GPT-2-Wechsel-Modelle für den Bootstrapping-Pool eines jeden Kreuzvalidierungs-Splits trainiert. Für jede Ensemblegröße wurden dann 1000 Ensembles zufällig aus dem Pool ausgewählt (mit Zurücklegen) und der RMSE des Ensembles für jeden Teil der Kreuzvalidierung ermittelt. Die Fehler wurden dann über die fünf Validierungs-Folds gemittelt, sodass für jede Ensemblegröße 5000 gemittelte RMSEs erhalten wurden.

Es konnte beobachtet werden, dass ein größeres Ensemble auch einen besseren RMSE induziert (siehe Abbildung 9). Dieser Effekt wurde allerdings ab einer Ensemblegröße von 20 deutlich schwächer. Auf der anderen Seite sorgte ein größeres Ensemble aber auch für ein stabileres Ergebnis.

Diese Tendenzen konnten für alle Ensemblekompositionen beobachtet werden, wobei das inhomogene Ensemble aus beiden Modellen das beste Ergebnis erzielte.

Entsprechend dieser Beobachtungen wurden im Rahmen der *Text Complexity Challenge DE 2022* zwei Arten der Einreichung durchgeführt; zum einen wurde ein Ensemble aus 340 GPT-2-Wechsel-Modellen eingereicht (*Submission A*) und zum anderen eines bestehend aus 100 GBERT- und GPT-2-Wechsel-Modellen (*Submission B*).

[18] *Bootstrapping* beschreibt hier den Vorgang, aus einem Pool von n paarweise unterschiedlichen Modellen eine Teilmenge von $m < n$ Modellen zu selektieren.

Insgesamt erreichte Submission A einen RMSE von 0.461 (transformiert[19]: 0.454) und Submission B einen RMSE von 0.442 (transformiert: 0.435).

Das Ensemble aus GBERT- und GPT-2-Wechsel-Modellen erreichte den zweiten Platz in dem Wettbewerb.

6 Fazit

Es wurde gezeigt, dass bereits existierende LLMs durchaus die Fähigkeit besitzen, die Komplexität deutscher Sätze zuverlässig vorherzusagen. Evaluiert wurden Ensembles mit verschiedenen Ensemblekompositionen und -größen bestehend aus GBERT- und GPT-2-Wechsel-Modellen. Dabei wurde klar, dass Ensembles aus beiden Modelltypen bessere Ergebnisse liefern, als homogene Ensembles.

[19] Die Ergebnisse wurden durch die Organisator:innen noch einmal linear transformiert. Siehe Kapitel 7.3 von ITU-T P.1401.

Literatur

[1] Sweta Agrawal und Marine Carpuat. „Controlling Text Complexity in Neural Machine Translation“. In: *Proc. 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. on Natural Language Processing, EMNLP-IJCNLP 2019*. Hrsg. von Kentaro Inui u. a. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, S. 1549–1564. DOI: 10.18653/v1/D19-1166. URL: https://doi.org/10.18653/v1/D19-1166.

[2] Lisa Beinborn, Torsten Zesch und Iryna Gurevych. „Towards Fine-Grained Readability Measures for Self-Directed Language Learning“. In: *Proc. SLTC 2012 workshop on NLP for CALL*. Linköping Electronic Conference Proceedings 80. 2012, S. 11–19. URL: https://ep.liu.se/ecp/080/002/ecp12080002.pdf.

[3] Tobias Bornheim, Niklas Grieger und Stephan Bialonski. „FHAC at GermEval 2021: Identifying German toxic, engaging, and fact-claiming comments with ensemble learning“. In: *CoRR* abs/2109.03094 (2021). arXiv: 2109.03094. URL: https://arxiv.org/abs/2109.03094.

[4] Tom Brown u. a. „Language models are few-shot learners“. In: *Advances in neural information processing systems* 33 (2020), S. 1877–1901.

[5] Branden Chan, Stefan Schweter und Timo Möller. „German's Next Language Model“. In: *Proc. 28th Int. Conf. on Computational Linguistics, COLING 2020*. Hrsg. von Donia Scott, Núria Bel und Chengqing Zong. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dez. 2020, S. 6788–6796. DOI: 10.18653/v1/2020.coling-main.598. URL: https://doi.org/10.18653/v1/2020.coling-main.598.

[6] Kevin Clark u. a. „What Does BERT Look at? An Analysis of BERT's Attention“. In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, S. 276–286. DOI: 10.18653/v1/W19-4828. URL: https://aclanthology.org/W19-4828.

[7] Kevyn Collins-Thompson. „Computational assessment of text readability: A survey of current and future research“. In: *International Journal of Applied Linguistics* 165.2 (2014), S. 97–135. ISSN: 0019-0829. DOI: https://doi.org/10.1075/itl.165.2.01col. URL: https://www.jbe-platform.com/content/journals/10.1075/itl.165.2.01col.

[8] Jacob Devlin u. a. „Bert: Pre-training of deep bidirectional transformers for language understanding“. In: *arXiv preprint arXiv:1810.04805* (2018).

[9] Rudolf Flesch. „How to write plain English“. In: *University of Canterbury. Available at http://www. mang. canterbury. ac. nz/writing_guide/writing/flesch. shtml.[Retrieved 5 February 2016]* (1979).

[10] Alex Graves. *Generating Sequences With Recurrent Neural Networks*. 2013. DOI: 10.48550/ARXIV.1308.0850. URL: https://arxiv.org/abs/1308.0850.

[11] Julia Hancke, Sowmya Vajjala und Detmar Meurers. „Readability Classification for German using Lexical, Syntactic, and Morphological Features“. In: *COLING 2012, 24th Int. Conf. on Computational Linguistics, Proc. Conf.: Technical Papers*. Hrsg. von Martin Kay und Christian Boitet. Mumbai, India: Indian Institute of Technology Bombay, Dez. 2012, S. 1063–1080. URL: https://aclanthology.org/C12-1065/.

[12] Sepp Hochreiter und Jürgen Schmidhuber. „Long short-term memory“. In: *Neural computation* 9.8 (1997), S. 1735–1780.

[13] Joseph Marvin Imperial. „Bert embeddings for automatic readability assessment“. In: *arXiv preprint arXiv:2106.07935* (2021).

[14] Bruce W Lee, Yoo Sung Jang und Jason Hyung-Jong Lee. „Pushing on text readability assessment: A transformer meets handcrafted linguistic features“. In: *arXiv preprint arXiv:2109.12258* (2021).

[15] Ilya Loshchilov und Frank Hutter. „Decoupled Weight Decay Regularization“. In: *7th Int. Conf. on Learning Representations, ICLR 2019*. New Orleans, LA, USA: OpenReview.net, Mai 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

[16] Matej Martinc, Senja Pollak und Marko Robnik-Šikonja. „Supervised and Unsupervised Neural Approaches to Text Readability“. In: *Computational Linguistics* 47.1 (März 2021), S. 141–179. DOI: 10.1162/coli_a_00398. URL: https://doi.org/10.1162/coli_a_00398.

[17] G Harry Mc Laughlin. „SMOG grading-a new readability formula“. In: *Journal of reading* 12.8 (1969), S. 639–646.

[18] Tomáš Mikolov u. a. „Statistical language models based on neural networks“. In: *Presentation at Google, Mountain View, 2nd April* 80.26 (2012).

[19] Benjamin Minixhofer, Fabian Paischer und Navid Rekabsaz. „WECHSEL: Effective initialization of subword embeddings for cross-lingual transfer of monolingual language models“. In: *CoRR* abs/2112.06598 (2021). arXiv: 2112.06598. URL: https://arxiv.org/abs/2112.06598.

[20] Babak Naderi u. a. „Automated Text Readability Assessment for German Language: A Quality of Experience Approach“. In: *11th Int. Conf. on Quality of Multimedia Experience QoMEX 2019*. Berlin, Germany: IEEE, Juni 2019, S. 1–3. DOI: 10.1109/QoMEX.2019.8743194. URL: https://doi.org/10.1109/QoMEX.2019.8743194.

[21] Babak Naderi u. a. „Subjective Assessment of Text Complexity: A Dataset for German Language“. In: *CoRR* abs/1904.07733 (2019). arXiv: 1904.07733. URL: http://arxiv.org/abs/1904.07733.

[22] Yann Ollivier, Corentin Tallec und Guillaume Charpiat. „Training recurrent networks online without backtracking“. In: *arXiv preprint arXiv:1507.07680* (2015).

[23] Alec Radford und Karthik Narasimhan. „Improving Language Understanding by Generative Pre-Training“. In: 2018. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[24] Alec Radford u. a. „Language Models are Unsupervised Multitask Learners“. In: 2019. URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[25] Luz Rello u. a. „Graphical schemes may improve readability but not understandability for people with dyslexia“. In: *Proceedings of the First Workshop on Predicting and Improving Text Readability for target reader populations*. 2012, S. 25–32.

[26] Julian Risch und Ralf Krestel. „Bagging BERT Models for Robust Aggression Identification“. In: *Proc. 2nd Workshop on Trolling, Aggression and Cyberbullying, TRAC@LREC 2020*. Hrsg. von Ritesh Kumar u. a. Marseille, France: European Language Resources Association (ELRA), Mai 2020, S. 55–61. URL: https://www.aclweb.org/anthology/2020.trac-1.9/.

[27] Anna Rogers, Olga Kovaleva und Anna Rumshisky. „A Primer in BERTology: What We Know About How BERT Works“. In: *Trans. Assoc. Comput. Linguistics* 8 (2020), S. 842–866. URL: https://transacl.org/ojs/index.php/tacl/article/view/2257.

[28] David E. Rumelhart, Geoffrey E. Hinton und Ronald J. Williams. „Learning representations by back-propagating errors“. In: *Nature* 323.6088 (Okt. 1986), S. 533–536. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0.

[29] Haşim Sak, Andrew Senior und Françoise Beaufays. „Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition“. In: *arXiv preprint arXiv:1402.1128* (2014).

[30] RJ Senter und Edgar A Smith. *Automated readability index*. Techn. Ber. Cincinnati Univ OH, 1967.

[31] Ilya Sutskever. „Training Recurrent Neural Networks“. AAINS22066. Diss. CAN, 2013. ISBN: 9780499220660.

[32] Sowmya Vajjala. „Trends, Limitations and Open Challenges in Automatic Readability Assessment Research“. In: *CoRR* abs/2105.00973 (2021). arXiv: 2105.00973. URL: https://arxiv.org/abs/2105.00973.

[33] Ashish Vaswani u. a. „Attention is all you need“. In: *Annual Conf. Neural Information Processing Systems 2017*. Long Beach, CA, USA, Dez. 2017, S. 5998–6008. arXiv: 1706.03762. URL: https://arxiv.org/abs/1706.03762.

[34] Zarah Weiß, Xiaobin Chen und Detmar Meurers. „Using Broad Linguistic Complexity Modeling for Cross-Lingual Readability Assessment“. In: *Proc.*

10th Workshop on Natural Language Processing for Computer Assisted Language Learning (NLP4CALL 2021). Linköping Electronic Conference Proceedings 177. 2021, S. 38–54. URL: https://aclanthology.org/2021.nlp4call-1.4.pdf.

[35] Zarah Weiß und Detmar Meurers. „Modeling the Readability of German Targeting Adults and Children: An empirically broad analysis and its cross-corpus validation“. In: *Proc. 27th Int. Conf on Computational Linguistics, COLING 2018*. Hrsg. von Emily M. Bender, Leon Derczynski und Pierre Isabelle. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, S. 303–317. URL: https://aclanthology.org/C18-1026/.

[36] Ronald Williams und Jing Peng. „An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories“. In: *Neural Computation* 2 (Sep. 1998). DOI: 10.1162/neco.1990.2.4.490.

[37] Angelika Wöllstein. „Syntax: lineare und hierarchische Gliederung. Mit Sprache über Sprache sprechen-grammatische Terminologie-eine Vorbemerkung“. In: *Sprachwissenschaft für das Lehramt*. Schöningh, 2014.