HAW HAMBURG

BACHELOR THESIS
Moritz Ohsten

NLP-Based Complex Word Identification in German

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE
WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Moritz Ohsten

NLP-Based Complex Word Identification in German

Bachelor thesis submitted for examination in Bachelor´s degree
in the study course *Bachelor of Science Informatik Technischer Systeme*
at the Department Computer Science
at the Faculty of Engineering and Computer Science
at University of Applied Science Hamburg

Supervisor: Prof. Dr. Marina Tropmann-Frick
Supervisor: Prof. Dr. Stefan Sarstedt

Submitted on: 04. Oct 2024

Moritz Ohsten

Thema der Arbeit

NLP-Based Complex Word Identification in German

Stichworte

Natürliche Sprachverarbeitung, Komplex, Wort, Deutsch, Transformer, BERT

Kurzzusammenfassung

Das Ziel der vorliegenden Arbeit ist es, zu untersuchen, wie gut sich transformerbasierte Modelle, insbesondere BERT, bei der Erkennung komplexer Wörter in der deutschen Sprache im Vergleich zu traditionellen Verfahren des maschinellen Lernens bewähren und wie sich diese Modelle optimieren lassen. Die Arbeit stützt sich auf die Ergebnisse und den Datensatz des CWI 2018 Shared Task, der im Rahmen dieser Arbeit zunächst um zusätzliche linguistische Merkmale erweitert wurde. Zur Bestimmung der Relevanz dieser Merkmale sowie als Benchmark diente ein Random Forest Classifier (RFC). Anschließend wurden verschiedene BERT-Modelle gegenübergestellt und Optimierungstechniken wie Hyperparameter-Optimierung und Datenaugmentation auf das beste Modell angewendet. Dieses Modell sowie die optimierten Varianten dieses Modells wurden den Top-Systemen des CWI 2018 Shared Task gegenübergestellt, um die Leistung zu vergleichen. Die Ergebnisse zeigen, dass die BERT-Modelle besser abschnitten als die traditionellen Machine-Learning-Verfahren aus dem CWI 2018 Shared Task. Besonders die Hyperparameter-Optimierung und die Einbeziehung linguistischer Merkmale trugen zu einer signifikanten Leistungssteigerung bei. Dies verdeutlicht das Potenzial transformerbasierter Modelle für die Aufgabe der CWI im Vergleich zu traditionellen Verfahren.

Moritz Ohsten

Title of Thesis

NLP-Based Complex Word Identification in German

Keywords

Natural Language Processing, Complex Word Identification, German, Transformer, BERT

Abstract

The aim of this thesis is to investigate how well transformer-based models, in particular BERT, perform in the recognition of complex words in German compared to traditional machine learning methods and how these models can be optimized. The work is based on the results and the data set of the CWI 2018 Shared Task, which was initially expanded to include additional linguistic features as part of this work. A Random Forest Classifier (RFC) was used to determine the relevance of these features and as a benchmark. Subsequently, different BERT models were compared and optimization techniques such as hyperparameter optimization and data augmentation were applied to the best model. This model and the optimized variants of this model were compared to the top systems of the CWI 2018 Shared Task to compare performance. The results show that the BERT models performed better than the traditional machine learning methods from the CWI 2018 Shared Task. Especially the hyperparameter optimization and the inclusion of linguistic features contributed to a significant increase in performance. This illustrates the potential of transformer-based models for the CWI task compared to traditional methods.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Problemstellung

Die Identifikation komplexer Wörter stellt ein bedeutendes Problem in der natürlichen Sprachverarbeitung dar und bildet den ersten Schritt bei der Vereinfachung von Texten für die Leserschaft. Trotz bemerkenswerter Fortschritte im Bereich des Natural Language Processing (NLP) bleibt die genaue Identifikation komplexer Wörter eine Herausforderung, insbesondere in Sprachen abseites vom Englischen. Diese Arbeit widmet sich der Untersuchung der Leistungsfähigkeit herkömmlicher maschineller Lernmethoden bei der Lösung der Aufgabe der Complex Woer Identification (CWI) und vergleicht diese mit modernen, transformerbasierten Ansätzen. Dabei wird der deutsche Datensatz des CWI Shared Task 2018 als Grundlage verwendet. Ziel dieser Arbeit ist es, die Effektivität der verschiedenen Methoden zur Identifikation komplexer Wörter im Deutschen zu bewerten und herauszufinden, welche Ansätze die besten Ergebnisse erzielen können.

1.2 Verwandte Arbeiten

In diesem Kapitel werden die bisherigen Ansätze, die sich mit CWI auseinandergesetzt haben, vorgestellt. Der **SemEval-2016 CWI Task** hatte das Ziel, Systeme zu entwickeln, die vorhersagen können, ob ein Nicht-Muttersprachler ein bestimmtes Wort in einem Satz verstehen würde (Paetzold und Specia, 2016). Hierfür wurde ein Datensatz erstellt, der aus dem CW Korpus (Shardlow, 2013b), LexMTurk Corpus (Horn u. a., 2014) und Simple Wikipedia Corpus (Kauchak, 2013) bestand. Diese Korpora wurden aus der Simple English Wikipedia[1] extrahiert. Von 400 Teilnehmern, deren Muttersprache nicht englisch ist, wurden die Wörter in dem Datensatz als komplex oder nicht komplex annotiert. Im

[1] https://simple.wikipedia.org/

Rahmen vom SemEval-2016 wurden 42 Systeme vorgestellt, die unterschiedliche Klassifikatoren und Typen von Merkmalen verwendeten, darunter lexikalische, morphologische, semantische und syntaktische Merkmale sowie psycholinguistische Maße. Die am häufigsten verwendeten Klassifikatoren unter den erfolgreichen Systemen waren Random Forest und Ensemble-Methoden, die eine Kombination aus lexikalischen und semantischen Merkmalen verwendeten. Die Ergebnisse des Tasks zeigten, dass Entscheidungsbäume und Ensemble-Methoden, die auf einer Kombination von lexikalischen, morphologischen und semantischen Merkmalen trainiert wurden, alle anderen maschinellen Lernverfahren, einschließlich neuronaler Ansätze, übertrafen. Darüber hinaus erwiesen sich einfache Features wie die Worthäufigkeit in Textkorpora als besonders effektiv.

Der **CWI Shared Task 2018** (Yimam u. a., 2018) war der zweite Task dieser Art. Hierfür wurde ein neuer Datensatz erstellt, der Texte aus den Bereichen News, Wikinews und Wikipedia umfasst. Neben dem Englischen wurden erstmals auch Deutsch und Spanisch für monolinguale Aufgaben berücksichtigt sowie ein französischer Testdatensatz für multilinguale Aufgaben bereitgestellt. Die Aufgabe der Annotatoren bestand darin, Wörter oder Phrasen zu markieren, die sie als schwierig empfanden, wobei es auch möglich war, Mehrwortausdrücke (MWEs) zu annotieren. Der CWI Shared Task 2018 gliederte sich in zwei Aufgaben:

- *Binäre Klassifikation*: Jedes Wort wurde entweder als komplex oder nicht komplex klassifiziert.
- *Probabilistische Klassifikation*: Jedes Wort erhielt einen Wahrscheinlichkeitswert, der auf der Anzahl der Annotatoren basierte, die das Wort als komplex einschätzten.

Für die binäre Klassifikationsaufgabe wurden der macro-averaged F1-Score und die Genauigkeit als Evaluierungsmetriken verwendet. Auch in diesem Task zeigte sich, dass Random Forest Klassifikatoren in Kombination mit Frequenz-, Längen- und semantischen Features besonders effektiv sind. Das Team TMU verwendete Arten von Frequenzfeatures basierend auf dem Lang-8-Lernerkorpus sowie dem allgemeinen Wikipedia- und WikiNews-Korpus (Kajiwara und Komachi, 2018). Die Features umfassten Wort- und Satzlänge sowie Wortfrequenz. Sie setzten Random Forest Klassifikatoren für die binäre Klassifikation ein. Das Team erreichte damit in der deutschen binären Klassifikation mit einem F1-Score von 0.7451 den ersten Platz und verfehlte die Baseline von 0.7546 nur knapp. SB@GU extrahierte morphologische, semantische und psycholinguistische Merkmale. Zu den Features zählten z.B. Wortlänge, Silbenanzahl, POS-Tag (Part-of-speech), Anzahl der Synsets in WordNet, und psycholinguistische Daten wie Häufigkeiten aus

dem British National Corpus. Sie verwendeten unter anderem Random Forest und Extra Trees Klassifikatoren (Alfter und Pilán, 2018) und erreichten in der deutschen binären Klassifikation den zweiten PLatz mit einem F1-Score von 0.7424.

Der **SemEval-2021 Shared Task** konzentrierte sich auf die Vorhersage der lexikalischen Komplexität von Wörtern und Mehrwortausdrücken mittels Regression (Shardlow u. a., 2021). Dabei gab es zwei Subtasks, wobei Subtask 1 nur einzelne Wörter behandelt und Subtask 2 Wörter und Mehrwortausdrücke. Der CompLex-Datensatz (Shardlow u. a., 2020), der für die Aufgabe verwendet wurde, bewertete Wörter auf einer Likert-Skala und umfasste Texte aus verschiedenen Domänen, wie der Bibel, Europarl und biomedizinischen Texten.

SemEval-2021 umfasste zahlreiche Systeme, die BERT als Grundlage für die Vorhersage der Wortkomplexität nutzten. Eines der erfolgreichsten Systeme war das JUST-BLUE-System (Yaseen u. a., 2021), das ein Ensemble aus BERT und RoBERTa verwendete und die beste Pearson-Korrelation (Sedgwick, 2012) für Subtask 1 erzielte. Ein weiteres leistungsstarkes System war DeepBlueAI (Pan u. a., 2021), das ebenfalls ein Ensemble aus RoBERTa und ALBERT verwendete und durch feingetunte Sprachmodelle eine hervorragende Leistung erbrachte. Dieses System erzielte die beste Pearson-Korrelation für Subtask 2 und war der Zweitplatzierte in Subtask 1.

Diese BERT-basierten Systeme zeigten, dass vortrainierte Sprachmodelle wie BERT, kombiniert mit Feature-Engineering oder Ensemble-Methoden, eine hohe Korrelation in der Vorhersage der Wortkomplexität erreichen konnten, wobei JUST-BLUE (Yaseen u. a., 2021) und DeepBlueAI (Pan u. a., 2021) zu den leistungsstärksten Systemen zählten.

1.3 Forschungsfragen

RQ1:
Wie gut eignen sich transformerbasierte Ansätze zur Erkennung komplexer Wörter in der deutschen Sprache im Vergleich zu herkömmlichen Ansätzen des maschinellen Lernens?
RQ2:
Welche Optimierungstechniken haben positiven Einfluss auf die Leistung der verschiedenen Modelle bei der Erkennung komplexer Wörter in deutschen Texten?

2 Theorie

2.1 Wortkomplexität

Was macht ein Wort eigentlich komplex? Diese Frage lässt sich nicht leicht beantworten, da es keine allgemeine Definition von Wortkomplexität gibt. Viele würden argumentieren, dass ein Wort als komplex gilt, wenn es eine bestimmte Länge, also Anzahl von Buchstaben oder Silben, überschreitet oder wenn es im alltäglichen Sprachgebrauch selten vorkommt (Gooding u. a., 2021). Allerdings ist diese Betrachtungsweise stark subjektiv, da jeder Mensch über einen individuellen Wortschatz verfügt und sich unterschiedlich ausdrückt.

Die Komplexität eines Wortes wird maßgeblich von der Zielgruppe beeinflusst. Wörter, die für einen Muttersprachler leicht verständlich sind, können für einen Zweitsprachler oder jemanden mit Sprachstörungen erhebliche Schwierigkeiten bereiten. Insbesondere die Seltenheit eines Wortes in einer Sprache ist ein wichtiger Faktor: Häufig verwendete Wörter werden in der Regel als weniger komplex wahrgenommen, während selten genutzte Wörter von den meisten Lesern als schwieriger empfunden werden (De Belder und Moens, 2010). Diese Korrelation zwischen Wortfrequenz und Vertrautheit ist ein zentrales Kriterium in der Bestimmung der Wortkomplexität.

Zusätzlich spielen auch andere Merkmale des Wortes eine Rolle. Die Wortlänge, gemessen in der Anzahl der Buchstaben, wird in vielen Lesbarkeitsformeln als Indikator für die Textkomplexität verwendet. Diese Annahme wurde durch zahlreiche Studien gestützt, die zeigten, dass längere Wörter oft als komplexer empfunden werden (Gooding u. a., 2021). Psycholinguistische Faktoren wie die Bildhaftigkeit und Konkretheit eines Wortes können ebenfalls die Verständlichkeit beeinflussen: Wörter, die leicht vorstellbar sind oder sich auf konkrete Objekte beziehen, sind in der Regel einfacher zu verstehen.

Es ist jedoch wichtig zu betonen, dass nicht alle Faktoren für jede Lesergruppe gleichermaßen relevant sind. Für Zweitsprachler ist die Wortfrequenz besonders entscheidend,

während für andere Gruppen, wie etwa Kinder oder Personen mit Aphasie, erlitten zum Beispiel durch einen Schlaganfall, die Länge eines Wortes oder bestimmte Buchstabenkombinationen eine größere Rolle spielen können (Knollman-Porter u. a., 2015). Diese Unterschiede verdeutlichen, dass die Klassifizierung eines Wortes als „komplex“ stark von den Eigenschaften des Lesers abhängt. Um Wortkomplexität sinnvoll bewerten zu können, muss man also stets die Zielgruppe und den Kontext der Textnutzung berücksichtigen.

2.2 CWI (Complex Word Identification)

Complex Word Identification (CWI) ist eine wichtige Teilaufgabe der Lexical Simplification (LS). LS hat das Ziel, komplexe Wörter in Texten durch einfachere Alternativen zu ersetzen, wobei Bedeutung und grammatikalische Korrektheit beibehalten werden, um so die Lesbarkeit zu verbessern (Shardlow, 2013a). Dies erleichtert Menschen mit geringer Lese- und Schreibkompetenz das Verständnis von Texten, zum Beispiel Nicht-Muttersprachlern oder älteren Menschen mit kognitiven Einschränkungen. CWI ist der erste Schritt in der LS-Pipeline und identifiziert Wörter, die für den Leser als zu komplex gelten. Diese Identifikation ist entscheidend, denn ungenaue Erkennungen können entweder dazu führen, dass zu viele komplexe Wörter unbeachtet bleiben und der Text somit schwer verständlich bleibt, oder dass unnötige Vereinfachungen vorgenommen werden, sodass sich der Kontext des Satzes stark verändert. Die LS-Pipeline kann in vier Hauptschritte unterteilt werden (Shardlow, 2014), welche in Abbildung 2.1 visualisiert sind:

Complex Word Identification: Hier werden alle komplexen Wörter in einem Text erkannt und klassifiziert.

Substitution Generation: Geeignete Substitutionen für die erkannten komplexen Wörter werden generiert.

Substitution Selection: Es werden diejenigen Substitutionen ausgewählt, die am besten in den Kontext passen.

Substitution Ranking: Die ausgewählten Substitutionen werden nach ihrer Eignung und Verständlichkeit sortiert.

CWI stellt dabei eine zentrale Herausforderung dar, da die Definition dessen, was ein komplexes Wort ist, oft nicht eindeutig ist. Wörter können je nach Kontext, Wortlänge,

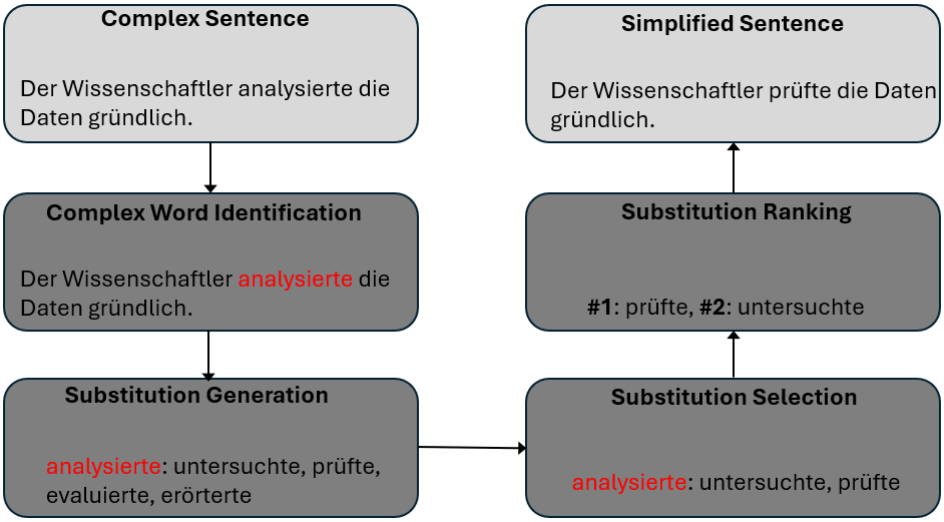


Abbildung 2.1: Schritte der Lexical Simplification Pipeline

Silbenanzahl oder auch Wortfrequenz als komplex gelten. So könnte man in einem Satz mit vielen Eigennamen wie „Honshu, Hokkaido, Shikoku und Kyushu“ entscheiden, dass diese Namen komplex sind, weil sie für den Leser unbekannt sind. In anderen Fällen könnte man sie als unwichtig für die Vereinfachung betrachten, da sie nicht durch alternative Begriffe ersetzt werden können (Shardlow, 2013a).

Die genaue Identifizierung von komplexen Wörtern ist auch deswegen schwierig, weil es keine universell anerkannten Methoden zur Bestimmung von Komplexität gibt. Manche Systeme stützen sich auf Häufigkeitsanalysen, andere wiederum auf lexikalische Eigenschaften wie Wortlänge oder Silbenanzahl. Eine fehlerhafte Identifikation kann dazu führen, dass zu viele Änderungen an einem Text vorgenommen werden, was seine ursprüngliche Bedeutung beeinträchtigen könnte. Viele Systeme schenken diesem Aspekt nicht genug Aufmerksamkeit, weshalb sich CWI als eigenständiges Forschungsgebiet etabliert hat. Forschungen aus anderen Arbeiten (Shardlow, 2014) zeigen, dass eine fehlerhafte Identifikation komplexer Wörter zu den häufigsten Fehlern in der gesamten LS-Pipeline führt. Dies unterstreicht die Notwendigkeit, den Prozess der CWI genauer zu untersuchen und zu verbessern, da die Qualität der nachfolgenden Schritte wesentlich davon abhängt.

2.2.1 Merkmale für CWI

Im Rahmen der CWI spielt die Auswahl geeigneter Features eine zentrale Rolle, um komplexe Wörter von einfachen zu unterscheiden. Die Qualität und die Vielfalt der verwendeten Merkmale beeinflussen maßgeblich die Leistung eines CWI-Systems. Um die Klassifikation von Wörtern als „komplex“ oder „nicht komplex“ zu ermöglichen, werden die Merkmale in verschiedene Kategorien unterteilt. Diese Kategorien umfassen lexikalische, morphologische, semantische und syntaktische Merkmale, die jeweils verschiedene Aspekte der Komplexität eines Wortes abbilden. Die folgenden Abschnitte stellen die wichtigsten Merkmale vor, die in der Literatur häufig verwendet werden, und erläutern, wie sie zur Identifizierung komplexer Wörter beitragen.

Lexikalische Merkmale
Lexikalische Merkmale beziehen sich auf die grundsätzlichen Eigenschaften von Wörtern, die aus der Wortform oder ihrer Häufigkeit im Text abgeleitet werden.

- *Worthäufigkeit:* Wie häufig ein Wort in einem bestimmten Korpus vorkommt. Komplexe Wörter treten in der Regel seltener auf als einfache Wörter.
- *Wortlänge:* Die Anzahl der Buchstaben eines Wortes. Längere Wörter sind oft schwieriger zu verstehen.
- *Silbenanzahl:* Wörter mit mehr Silben gelten in der Regel als komplexer.

Morphologische Merkmale
Morphologische Merkmale beziehen sich auf die Struktur eines Wortes, insbesondere darauf, wie es aus kleineren Einheiten (Morphemen) zusammengesetzt ist.

- *Anzahl der Morpheme:* Wörter, die aus mehreren Morphemen (z.B. Präfixen, Suffixen) bestehen, werden oft als komplexer eingestuft.
- *Wortableitung*: Wörter, die von anderen Wortformen abgeleitet sind, wie z.B. Adjektive oder Substantive, die aus Verben gebildet wurden (z.B. „Entwicklung“ von „entwickeln“).

Semantische Merkmale
Semantische Merkmale beziehen sich auf die Bedeutung von Wörtern und deren Beziehung zu anderen Wörtern im Text.

- *Polysemie(Mehrdeutigkeit):* Wörter mit mehreren Bedeutungen können für Leser schwieriger zu verstehen sein, da der Kontext die Bedeutung klären muss.
- *Synonymhäufigkeit*: Wörter, die viele Synonyme haben, werden tendenziell als einfacher angesehen, da der Leser eher eine Bedeutung erkennen kann.
- *Konzeptebene*: Abstrakte Begriffe sind in der Regel schwieriger zu verstehen als konkrete Begriffe.
- *Semantische Ähnlichkeit:* Wie ähnlich ein Wort in einem Text/Satz anderen Wörtern ist, was auf seine Verständlichkeit hinweist (zum Beispiel Kosinus-Ähnlichkeit, Kapitel 4.1.5).

Syntaktische Merkmale
Syntaktische Merkmale beziehen sich auf die Rolle eines Wortes in einem Satz und dessen grammatikalische Struktur.

- *Wortart (Part of Speech):* Substantive, Verben, Adjektive und Adverbien können je nach Kontext unterschiedlich schwierig sein. Substantive sind oft als komplexer angesehen.
- *Satzposition:* Wörter an bestimmten Stellen im Satz (z.B. am Anfang oder Ende) können für Leser leichter oder schwerer verständlich sein.
- *Satzlänge:* Längere Sätze enthalten oft komplexere Strukturen und machen das Verständnis schwieriger.
- *Satzstruktur:* Komplexe Sätze, wie z.B. solche mit vielen Nebensätzen, können die Verständlichkeit von Wörtern im Kontext erschweren.

2.3 Metriken zur Modellbewertung

In vielen Bereichen des maschinellen Lernens ist die Evaluierung von Modellen mithilfe geeigneter Metriken ein zentraler Schritt, um ihre Leistungsfähigkeit zu bewerten. Ein Großteil dieser Modelle fungiert jedoch als sogenannte „Blackbox“, was bedeutet, dass ihre interne Logik und Funktionsweise für den Nutzer nicht ersichtlich sind (Carvalho u. a., 2019). Dies macht die Einführung von Metriken notwendig, die nicht nur die Modellleistung quantifizieren, sondern auch zur Erklärung und Nachvollziehbarkeit beitragen.

Zu den gebräuchlichsten Metriken zur Leistungsbewertung zählen Accuracy, Precision, Recall und der F1-Score. Zur Berechnung dieser Metriken dient die Konfusionsmatrix als Grundlage, die eine Übersicht über die tatsächlichen und vorhergesagten Klassifikationen eines Modells bietet.

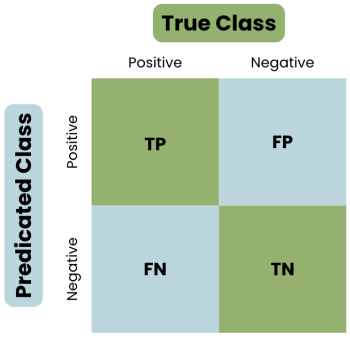


Abbildung 2.2: Konfusionsmatrix. (Ahmed, 2023)

Anhand der Abbildung 2.2 lässt sich die Matrix wie folgt erklären:

Die Matrix ist in vier Quadranten aufgeteilt, die die möglichen Ergebnisse einer binären Klassifikation darstellen. Die Spalten repräsentieren die tatsächliche Klasse (True Class), unterteilt in „Positive“ und „Positive“. Die Zeilen zeigen die vorhergesagte Klasse (Predicted Class), ebenfalls in „Positive“ und „Positive“ unterteilt. Die vier Quadranten bedeuten:

- *TP (True Positive)*: Korrekt als positiv klassifiziert
- *FP (False Positive*): Fälschlicherweise als positiv klassifiziert
- *FN (False Negative)*: Fälschlicherweise als negativ klassifiziert
- *TN (True Negative)*: Korrekt als negativ klassifiziert

Die diagonalen Felder (TP und TN) zeigen die korrekten Vorhersagen, während die anderen beiden Felder (FP und FN) die Fehler des Modells darstellen. Diese Matrix hilft bei der Beurteilung, wie gut ein Modell zwischen den Klassen unterscheiden kann. Würde ein Modell im Hinblick auf CWI ein Wort in einem Satz, welches als „nicht komplex“ annotiert wurde, als „komplex“ klassifizieren, würde man diese Ergebnis im oberen rechten Quadranten (FP) einordnen. Anhand der Abbildung 2.2 lassen sich die Metriken erklären:

Accuracy (Genauigkeit) misst den Anteil aller korrekten Vorhersagen an der Gesamtzahl der Vorhersagen. Sie gibt an, wie oft das Modell insgesamt richtig liegt. Die Formel für Accuracy lautet:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

Precision (Präzision) beschreibt den Anteil der korrekt als positiv klassifizierten Fälle an allen als positiv klassifizierten Fällen. Diese Metrik zeigt, wie zuverlässig die positiven Vorhersagen des Modells sind. Die Formel für Precision ist:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.2}$$

Recall (Sensitivität) gibt den Anteil der korrekt als positiv klassifizierten Fälle an allen tatsächlich positiven Fällen an. Diese Metrik misst, wie gut das Modell alle positiven Fälle erkennt. Die Formel für Recall lautet:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.3}$$

Der **F1-Score** ist das harmonische Mittel von Precision und Recall. Er bietet eine ausgewogene Bewertung und ist besonders nützlich bei unausgeglichenen Datensätzen. Die Formel für den F1-Score ist:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.4}$$

Der Macro F1-Score ist der Durchschnitt der F1-Scores für jede Klasse in einem Multi-Klassen-Problem. Er gibt jedem F1-Score die gleiche Gewichtung, unabhängig von der Klassengröße, was besonders nützlich bei unausgeglichenen Datensätzen ist. Die Formel für den Macro F1-Score lautet:

$$\text{Macro F1-Score} = \frac{1}{N} \sum_{i=1}^{N} \text{F1-Score}_i \tag{2.5}$$

N ist dabei die Anzahl der Klassen ist. Der Macro F1-Score ist wichtig in Szenarien, wo die Leistung des Modells über alle Klassen hinweg konsistent sein soll, unabhängig von der Häufigkeit jeder Klasse im Datensatz. Er hilft dabei, ein ausgewogenes Bild der Modellleistung zu erhalten, insbesondere wenn einige Klassen unterrepräsentiert sind.

2.4 Transformer-Sprachmodelle

Mit der Einführung von Transformer Sprachmodellen durch Vaswani u. a. (2017) mit dem Paper „Attention is all you need“ wurde die Art und Weise revolutioniert, wie Modelle Sprache verarbeiten und verstehen. Die Hauptaufgabe der Transformer Sprachmodelle ist die Sequenztransduktion. Hierbei geht es um die Umwandlung einer Sequenz von Eingabedaten in eine Sequenz von Ausgabedaten. Dazu gehören Aufgaben wie Textübersetzung, Zusammenfassen von Texten, Spracherkennung, Text-to-Speech und vieles mehr. Transformer lösten damit die RNNs (Recurrent Neuronal Networks) ab, die bis dato führend auf diesem Gebiet waren. Der Unterschied zwischen RNNs und FNNs (Feed Forward Neural Network) liegt darin, wie Informationen durch das Netz weitergegeben werden. RNNs besitzen Rückkopplungen, die es ihnen erlauben, Informationen aus vorherigen Zeitschritten zu berücksichtigen (Zhang u. a., 2019).

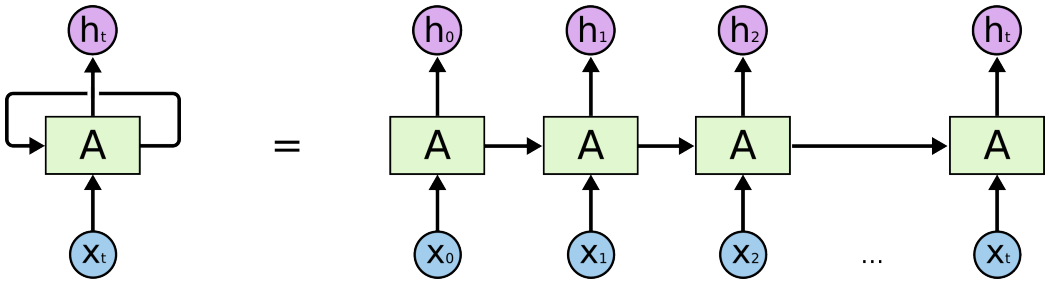


Abbildung 2.3: Rekurrentes Netz entrollt. (Olah, 2015)

Dabei werden die Informationen der bisherigen Eingaben in einem „Hidden State“ gespeichert. Er fungiert sozusagen als Gedächtnis. Abbildung 2.3 zeigt einen Teil eines RNNs. Vorstellen kann man sich ein RNN, wie auf der rechten Seite dargestellt, auch entrollt als quasi FNN. Diese Architektur bringt allerdings auch einige Nachteile mit sich:
Vanishing/Exploding Gradient: Bei der Backpropagation Through Time (BPTT), die zur Fehlerminimierung in RNNs verwendet wird, können die Gradienten über viele Zeitschritte hinweg durch die wiederholte Matrixmultiplikation sehr klein werden oder gar verschwinden (Chen, 2018). Das führt zum einen dazu, dass **Langzeitabhängigkeiten** schlecht erlernt werden. Genauer gesagt bedeutet es, dass oft relevante Informationen am Anfang eines Textes „vergessen“ werden, wenn diese Netze sich mit späteren Teilen des Textes befassen. Andererseits können die Gradienten durch große Werte in der Matrixmultiplikation auch explodieren, was zu instabilen Trainingsprozessen führt.
Performanz: Das Training ist dadurch, dass RNNs Daten sequeziell verarbeiten, also beispielsweise Wort für Wort, sehr zeitaufwendig und schwer parallelisierbar.

LSTMs (Long Short Term Memory), vorgestellt von Hochreiter und Schmidhuber (Hochreiter und Schmidhuber, 1997), sollten insbesondere die Probleme der Langzeitabhängigkeiten und Vanishing Gradients lösen. Durch das Verwenden von „Gates“ (Input-, Output- und Forget-Gates) innerhalb ihrer Zellen steuern sie, welche Informationen behalten, hinzugefügt oder vergessen werden. Dadurch können LSTMs Langzeitabhängigkeiten zwar besser erfassen als normale RNNs, kommen bei sehr langen Sequenzen jedoch auch hier an ihre Grenzen, da die Informationen ebenfalls in einem Hidden-State gespeichert werden.

Im Gegensatz zu RNNs und LSTMs ermöglicht die Transformer-Architektur eine parallele Verarbeitung von Daten. Dies wird durch den Self-Attention-Mechanismus erreicht, der es dem Modell erlaubt, den Kontext jedes Wortes in einem Satz unabhängig von seiner Position im Satz zu erfassen. Wie in Abbildung 2.4 veranschaulicht, kann man den Transformer in zwei größere Teile unterteilen: einen Encoder- und einen Decoderteil, welche sich wiederum aus mehreren Layern zusammensetzen.

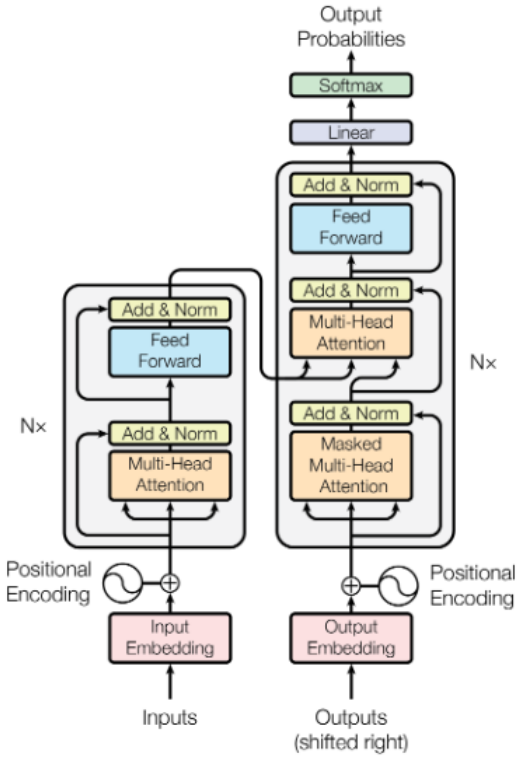


Abbildung 2.4: Transformer Architektur. (Vaswani u. a., 2017)

Der **Input Emmbedding Layer** konvertiert Wörter eines Satzes in Word Embeddings. Das ist eine Methode, um Wörter als Vektoren so zu repräsentieren, dass ähnliche Wörter auch ähnliche Vektoren besitzen. Zum Beispiel soll das Wort „cat“ nah an dem Word „dog“ als Vektor liegen und weit entfernt von dem Wort „car“. Auch wenn „cat“ und „car“ sich nur in einem Buchstaben unterscheiden, besitzen sie einen unterschiedlichen Kontext. Der Embedding Layer produziert einen 512 dimensionalen Vektor für jedes Wort eines Satzes. Dieser wird der nächsten Komponente des Transformers übergeben (Jalammar, 2018).
Der Encoder weiß nicht, in welcher Reihenfolge sich die Wörter befinden. Da die Wortreihenfolge aber einen großen Einfluss auf die Bedeutung eines Satzes hat, braucht der Encoder diese Information. Die Hauptaufgabe des **Positioning Encoding Layer** liegt folglich darin, Informationen über die Positionen der Wörter, den Embeddings hinzuzufügen, bevor diese dem Encoder übergeben werden. Der Kern des Transformers ist der Self-Attention-Mechanismus. Dieser Mechanismus ermittelt die Bedeutung jedes Wortes im Kontext aller anderen Wörter in einem Satz, um eine umfassende Repräsentation des gesamten Satzes zu erstellen. Jedes Wort wird zunächst in einen Embedding-Vektor umgewandelt. Aus jedem dieser Vektoren werden drei separate Vektoren erzeugt: Query (Q), Key (K) und Value (V).

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.6}$$

Die Query repräsentiert die Anfrage des Modells und kann beispielsweise ein Wort sein, für das relevante Informationen von anderen Wörtern im Satz gesucht werden. Die Keys repräsentieren die Wörter, die mit der Query verglichen werden, um festzustellen, wie stark sie miteinander in Beziehung stehen. Diese Beziehung wird durch die Berechnung von Ähnlichkeiten (oft mittels des Skalarprodukts) zwischen Query und Key ermittelt. Values sind die Vektoren, die die relevanten Informationen tragen und am Ende gewichtet werden. Wenn ein Key gut zur Query passt, trägt der entsprechende Value signifikant zur endgültigen Ausgabe bei. Alle drei Vektoren haben eine Dimension von 64, um die Berechnungen effizienter zu gestalten.

Die Query-Vektoren werden mit den Key-Vektoren multipliziert, um Aufmerksamkeits-Scores zu berechnen. Diese Scores werden durch die Quadratwurzel der Dimension (64)

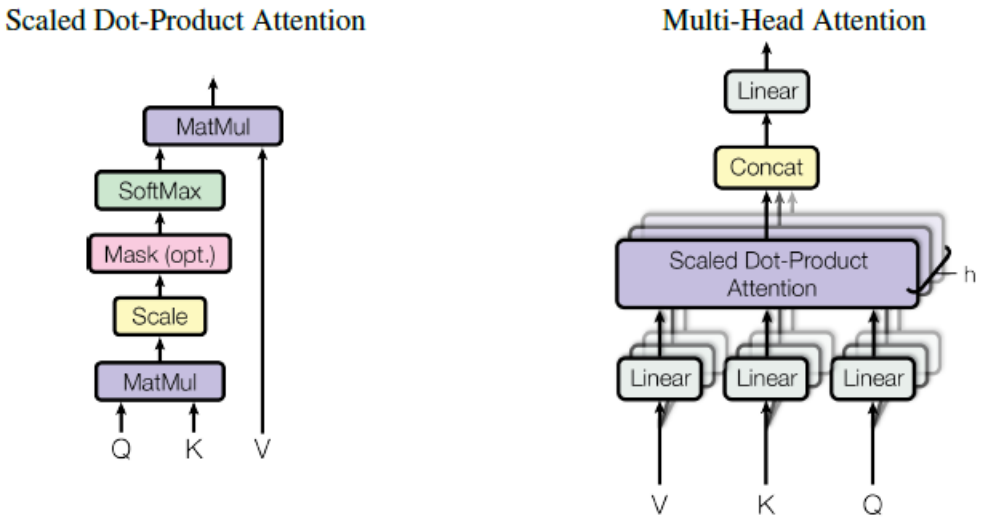


Abbildung 2.5: Scaled Dot-Product Attention (links) und Multi-Head Attention (rechts). (Vaswani u. a., 2017)

geteilt, um stabilere Gradienten zu erzielen. Anschließend werden die Scores mit der Softmax-Funktion normalisiert, sodass alle Werte zwischen 0 und 1 liegen und sich zu 1 summieren. Diese normalisierten Scores werden dann genutzt, um die Value-Vektoren zu gewichten, und die gewichteten Werte werden summiert, um die endgültige Ausgabe des Self-Attention-Layers für jedes Wort zu erhalten. In Abbildung 2.6 ist dieser Vorgang an einem Beispiel sichtbar. In der praktischen Implementierung werden die Berechnungen auf Matrizenebene durchgeführt: Die Embeddings der Eingabedaten liegen als Matrizen vor und werden dann durch die Gewichtsmatrizen in die Query-, Key- und Value-Matrizen umgewandelt. Diese Gewichtsmatrizen werden während des Trainings optimiert (Jalammar, 2018).

In der Arbeit von Vaswani u. a. (2017) wird nicht nur ein einzelner Self-Attention-Layer verwendet, sondern es kommen mehrere sogenannte Attention-Heads parallel zum Einsatz, genauer gesagt acht. Dieses Konzept wird als Multi-Head Attention bezeichnet. Durch den Einsatz mehrerer Attention-Heads kann das Modell verschiedene Aspekte der Eingabe gleichzeitig berücksichtigen, da jeder Head durch seine eigenen Gewichtsmatrizen unterschiedliche Teile der Sequenz in den Fokus nimmt. Dadurch kann das Modell feinere Beziehungen zwischen den Tokens erkennen und die Generalisierung verbessern. Da der anschließende Feed-Forward-Layer im Encoder, welcher die Aufgabe hat, jede Position in der Sequenz unabhängig voneinander weiter zu verarbeiten und durch nicht-lineare

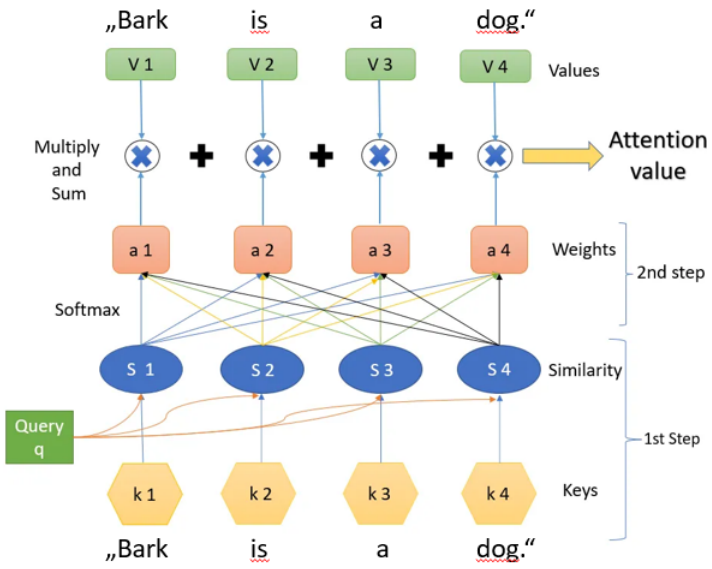


Abbildung 2.6: Attention-Meschanismus Beispiel. (Arjun, 2022)

Transformationen komplexere Zusammenhänge zu modellieren, jedoch nur eine einzige Eingabematrix erwartet, werden die Ausgaben der einzelnen Attention-Heads zunächst konkateniert und anschließend mit einer neuen Gewichtsmatrix multipliziert, die während des Trainings optimiert wird. Dies ermöglicht es, die verschiedenen Informationsquellen der Multi-Head Attention effizient zusammenzuführen (Jalammar, 2018).

Der Encoder gibt für jedes Wort einen Vektor aus, der nicht nur seine Bedeutung und die Position erfasst, sondern auch seine Interaktion mit anderen Wörtern mittels der Multi-Head-Attention. Diese Vektoren werden in Keys und Values transformiert und dem Decoder übergeben.

Ähnlich wie auf der Encoder-Seite verwendet der Decoder einen Embedding-Layer sowie eine Positional-Encoding-Schicht, um die Position der Tokens in der Sequenz zu erfassen. Der wesentliche Unterschied liegt jedoch im ersten Attention-Layer. Da der Decoder während des Trainings die vollständige Zielsequenz als Eingabe erhält, erfolgt im ersten Attention-Layer eine Maskierung. Vor der Anwendung der Softmax-Funktion werden alle Tokens, die im Index hinter dem aktuell betrachteten Token liegen, mit dem Wert „-inf“ maskiert. Dieser Maskierungsprozess stellt sicher, dass der Decoder bei der Berechnung der Self-Attention ausschließlich auf frühere Tokens zugreifen kann und keine Informationen über zukünftige Tokens berücksichtigt, die noch nicht generiert wurden.

Der zweite Attention-Layer im Decoder-Teil ist der Encoder-Decoder-Attention-Layer. Hierbei stammen die Keys und Values aus den Ausgaben des Encoders, während die Queries vom Output des vorangegangenen Masked-Multi-Head-Attention-Layers des Decoders generiert werden. Das hilft dem Decoder, sich auf passende Stellen der Eingabesequenz zu konzentrieren.

Durch den Self-Attention-Mechanismus meidet der Transformer gezielt die Herausforderungen, die bei RNNs häufig auftreten. Der effiziente Multi-Head-Attention-Mechanismus ermöglicht es dem Modell, Informationen über die gesamte Sequenz hinweg parallel zu verarbeiten, unabhängig von der Distanz zwischen den Tokens. Dies führt dazu, dass das Problem der Langzeitabhängigkeiten weitgehend gelöst wird, da jedes Token sofort mit allen anderen in der Sequenz interagieren kann. Im Gegensatz zu RNNs, die rekursiv und schrittweise über die Zeit arbeiten, entfällt im Transformer die Gefahr der exponentiellen Abschwächung oder Verstärkung der Gradienten, was das Problem des Vanishing bzw. Exploding Gradient verhindert.

Zudem bietet der Transformer durch seine Architektur die Möglichkeit, alle Positionen der Eingabesequenz gleichzeitig zu verarbeiten, was sowohl beim Training als auch bei der Inferenz zu erheblichen Leistungssteigerungen führt. Diese Parallelisierbarkeit stellt einen bedeutenden Vorteil gegenüber sequentiellen Modellen dar, die aufgrund ihrer Schritt-für-Schritt-Verarbeitung keine effiziente Nutzung moderner Hardware wie GPUs oder TPUs erlauben. Der Transformer überwindet damit nicht nur das Vanishing-Gradient-Problem, sondern auch die Einschränkungen durch fehlende Parallelisierung, was ihn zu einer äußerst leistungsstarken und skalierbaren Architektur macht.

Transformer-Sprachmodelle lassen sich in drei Haupttypen unterteilen: Encoder-only, Decoder-only und Encoder-Decoder Modelle. Sie unterscheiden sich hibsichtlich ihrer Architektur und Eignung für verschiedene NLP-Aufgaben. **Encoder-only Modelle** nutzen nur den Encoder-Teils des Transformers. Sie werden oft ach als auto-encoding Modelle bezeichnet. Sie besitzen einen bidirektionalen Attention Mechnismus, der es ihnen erlaubt, jederzeit auf alle Wörter in einem Satz gleichzeitig zuzugreifen und können den Kontext eines Wortes in dem Bezug auf die vorhergehenden als auch die nachfolgenden Wörter berücksichtigen. Das Pretraining solcher Modelle beinhaltet oft das Maskieren von Wörtern in einer Eingabesequenz und das anschließende Rekonstruieren der originalen Sequenz. Zu den Aufgaben zählen vor allem solche, die ein Verständnis der gesamten Eingabesequenz erfordern. Dazu gehören zum Beispiel Textklassifikation, Named Enti-

ty Recognition oder Question-Answering. Beispielmodelle hierfür sind sämtliche Modelle der BERT-Familie (Bidirectional Encoder Representations from Transformers) (HuggingFace, 2024).

Decoder-only Modelle, oft auch autoregressive Modelle genannt, nutzen ausschließlich den Decoder-teil des Transformers. Sie können in jedem Verarbeitungsschritt nur auf vorangegangene Wörter der Eingabesequenz zugreifen und sagen das nächste Wort aus diesen Informationen vorher. Auf diese Vorhersagen des nächsten Wortes in einem Satz bezieht sich typischerweise auch das Pretraining. Sie eignen sich besonders gut für Aufgaben der Textgenerierung. Dazu gehören Textübersetzung und die Zusammenfassung von Texten. Bekannte Vertreter diese Modelle sind GPT, GPT-2, GPT-3 (HuggingFace, 2024).

Encoder-Decoder-Modelle kombinieren beide vorherigen Ansätze und benutzen folglich sowohl Encoder als auch Decoder. Sie werden deshalb auch Sequence-to-sequence models genannt. Sie eignen sich gut für Aufgaben, die auf der Generierung neuer Sätze basieren und von einer gegebenen Eingabe abhängen. Typische Anwendungsbereiche sind Textzusammenfassung, Maschinelle Übersetzung und generative question-answering. Zu genannten Modellen gehören zum Beispiel BART, mBART oder T5 (HuggingFace, 2024).

Für die Aufgabe der Complex Word Identification (CWI) ist der Encoder-only Ansatz, insbesondere in Form von Modellen wie BERT, besonders gut geeignet. Dies liegt vor allem an der Fähigkeit dieser Modelle, den bidirektionalen Kontext eines Wortes in der Eingabesequenz zu analysieren. Diesen Vorteil bringen Decoder-only Modelle nicht mit sich. CWI erfordert es, den Schwierigkeitsgrad eines Wortes in seinem Kontext zu beurteilen, da die Komplexität eines Wortes oft auch von der umgebenden Syntax und Semantik abhängt.

Da auch in dieser Arbeit Encoder-only-Lösungsasätze betrachtet werden, wird im nächsten Kapitel BERT vorgestellt und auf die verbleibenden zwei Haupttypen nicht weiter eingegangen.

2.4.1 Transferlernen

Menschen sind besonders geschickt darin, Wissen von einer Domäne auf eine ähnliche Aufgabe in einer anderen Domäne zu übertragen. Ein anschauliches Beispiel ist das Erlernen von Musikinstrumenten. Eine Person, die bereits sehr begabt im Gitarrenspiel ist, wird wahrscheinlich weniger Schwierigkeiten haben, das Klavierspielen zu erlernen als jemand, der keine musikalische Vorerfahrung hat. Obwohl sich die Techniken des Klavier- und Gitarrenspiels stark unterscheiden, besitzt der Gitarrenspieler ein ausgeprägtes Taktgefühl und die Fähigkeit, Noten zu lesen, was den Lernprozess wesentlich erleichtert. Ähnlich wie beim menschlichen Lernen lässt sich dieses Prinzip auch auf Modelle im maschinellen Lernen übertragen. Das Zitat „Transfer learning aims at improving the performance of target learners on target domains by transferring the knowledge.“ (Zhuang u. a., 2021) beschreibt dies treffend. Transfer Learning ist ein Verfahren im maschinellen Lernen, bei dem Wissen, das aus einer Aufgabe (der sogenannten Quell-Domäne) erworben wurde, genutzt wird, um die Leistung in einer anderen, aber verwandten Aufgabe (der Ziel-Domäne, zum Beispiel CWI) zu verbessern.

2.4.2 BERT

BERT (Bidirectional) Encoder Representations from Transformers) ist ein Sprachmodell, das auf dem Transformer-Modell von Vaswani et al. (2017) basiert. Es war eines der ersten Transformer-Modelle, das bidirektional trainiert wurde, sodass es den Kontext eines Wortes sowohl von links als auch von rechts gleichzeitig berücksichtigt (Devlin u. a., 2018). Frühere Modelle verarbeiteten Sätze in einer unidirektionalen Weise, indem sie den Text entweder von links nach rechts oder von rechts nach links sequenziell durchgingen. BERT hingegen nutzt nur den Encoder-Teil des Transformers.

Wie bei anderen modernen Sprachmodellen setzt BERT auf Transferlernen. Das Framework ist in zwei Phasen unterteilt: Pretraining und Finetuning. Beim Pretraining wird das Modell auf großen, allgemeinen Textkorpora trainiert, während beim Finetuning spezifische Aufgaben wie Textklassifikation, Fragebeantwortung oder Named Entity Recognition optimiert werden. Für die Tokenisierung verwendet BERT die WordPiece Embeddings (Wu et al., 2016) mit einem Vokabular von etwa 30.000 Token. Diese Methode teilt Wörter in Teilwörter auf, was eine bessere Verarbeitung von seltenen Wörtern ermöglicht. So kann das Modell sowohl häufige Begriffe als auch unbekanntere Begriffe

effizient verarbeiten. Das Pretraining wurde auf zwei großen Datensätzen durchgeführt: dem BookCorpus (Zhu, 2015) und English Wikipedia (2,5 mrd Wörter). Es umfasste zwei Aufgaben: Masked Language Modeling (MLM) und Next Sentence Prediction (NSP).

Beim MLM werden 15 % der Token einer Sequenz zufällig ausgewählt und maskiert. Das Modell muss dann diese maskierten Wörter anhand des Kontexts vorhersagen. Um die Diskrepanz zwischen Pretraining und Finetuning zu verringern, da beim Finetuning, zum Beispiel im Falle einer Textklassifikation, keine [MASK]-Tokens vorkommen, wird nicht immer ein [MASK]-Token verwendet. In 80 % der Fälle wird das Wort mit [MASK] ersetzt, in 10 % der Fälle durch ein zufälliges Token und in den restlichen 10 % bleibt das ursprüngliche Wort unverändert. Diese Technik verbessert die Generalisierungsfähigkeit des Modells, da es lernt, in verschiedenen Kontexten vorherzusagen, ohne sich zu sehr auf den [MASK]-Token zu verlassen.

Um auch Beziehungen zwischen Sätzen zu verstehen, wird BERT auf dem NSP -Task trainiert, bei dem es vorhersagen muss, ob ein Satz B auf einen Satz A folgt. Zwei Sätze werden durch den speziellen [SEP]-Token getrennt und dem Modell als Eingabe gegeben. Das Modell soll dann entscheiden, ob B tatsächlich auf A folgt oder nicht. Dafür wurden die Sätze aus dem Trainingsdatensatz mit „IsNext“ oder „NotNext“ annotiert Devlin u. a. (2018). Diese Aufgabe hilft BERT, Beziehungen zwischen Sätzen zu erfassen, was für Aufgaben wie Question-Answering wichtig ist.

Die Kombination dieser beiden Pretraining-Aufgaben ermöglicht es BERT, sowohl Beziehungen zwischen Wörtern, als auch zwischen Sätzen zu lernen.

3 Datensatz

Der Datensatz, welcher für die folgenden Experimente verwendet wird, ist der deutsche Datensatz des CWI Shared Task 2018 (Yimam u. a., 2018). Er besteht aus einer Sammlung von rund 978 Sätzen, extrahiert aus Wikipedia Artikeln (Yimam u. a., 2017) und ist bereits in Trainings- Validierungs- und Testdaten unterteilt (siehe dazu Tabelle 3.1).

Training	Dev	Test
6.151	795	959

Tabelle 3.1: Aufteilung des Datensatzes

Der Trainings- und Testdatensatz weist eine einheitliche Struktur auf. Jedes Sample besteht aus elf Spalten (siehe Abbildung 2 in Anhang A.2) illustriert die Struktur am Beispiel einer Stichprobe aus dem Datensatz.

Die erste Spalte gibt die HIT (Human Intelligence Task) ID des Satzes an. Jeder HIT umfasst stets 5 bis 10 Sätze, die gemeinsam einen Paragraphen bilden, welcher von zehn Annotatoren bewertet wurde. Die zweite Spalte enthält den Satz selbst. Die dritte und vierte Spalte spezifizieren den Beginn und das Ende des Offsets im Satz, in dem sich das zu klassifizierende Wort befindet. In der fünften Spalte ist das Zielwort aufgeführt. Dieses Zielwort kann entweder ein einzelnes Wort oder eine Wortsequenz darstellen, umfasst jedoch maximal 50 Satzzeichen. Da es sich bei mehr als 92% der annotierten Wörter um einzelne Wörter handelt (Yimam u. a., 2017), wird im Verlauf immer das Wort „Zielwort“ für das zu klassifizierende Wort verwendet, auch wenn es sich um eine Wortsequenz handelt. Die folgenden beiden Spalten zeigen die Anzahl der Annotatoren, deren Muttersprache Deutsch ist, sowie derer, deren Muttersprache eine andere Sprache ist. In der achten Spalte wird die Anzahl der deutschsprachigen Annotatoren aufgeführt, die das Zielwort als komplex bewerteten. Die neunte Spalte gibt die Anzahl der nicht-deutschsprachigen Annotatoren an, die das Zielwort als komplex einstuften. Die zehnte Spalte dokumentiert die binäre Klassifikation: Ein Wort wird als komplex eingestuft, wenn es von mindestens

einem Annotator als solches bewertet wurde. Schließlich beschreibt die elfte Spalte die probabilistische Klassifikation, die folgendermaßen berechnet wird:

$$P_{\text{komplex}} = \frac{\text{Anzahl der Annotatoren, die das Wort als komplex markiert haben}}{\text{Gesamtanzahl der Annotatoren}} \tag{3.1}$$

Für die Experimente in den folgenden Kapiteln wird ausschließlich die binäre Klassifikation verwendet. Daher sind für die weiteren Analysen nur noch die Spalten von Interesse, die den Satz, das Zielwort, die binäre Klassifikation sowie den Anfang und das Ende des Offsets enthalten, da diese Information später zur Berechnung der Wortlänge verwendet werden.

4 Methodik

4.1 Integration linguistischer Merkmale in den Datensatz

Um aufwändige Feature-Engineering-Prozesse zu vermeiden, ist es das Ziel, möglichst wenige Merkmale mit hohem Einfluss zu verwenden. In den folgenden Kapiteln werden die in dieser Arbeit verwendeten Merkmale näher erläutert und beschrieben, wie der Datensatz um diese erweitert wurde. Durch die Experimente soll unter anderem herausgefunden werden, welche Rolle die einzelnen Merkmale für die Klassifikation spielen.

4.1.1 Wortlänge

Für jedes Zielwort im Datensatz wurde die Anzahl der Buchstaben berechnet. Dafür wurde die Differenz der beiden Offsets errechnet. Die Annahme ist, dass ein Wort für einen Lerner einer Sprache umso komplexer ist, desto länger das Wort ist.

4.1.2 Silbenanzahl

Dieses Merkmal wurde mit demselben Hintergedanken ausgewählt, wie die Wortlänge. Je mehr Silben ein Wort hat desto komplexer könnte es für den Leser erscheinen. Im Gegensatz zur Wortlänge, wo ein Wort tatsächlich desto länger ist, je mehr Buchstaben es hat, muss ein Wort mit mehr Silben nicht gleich länger sein als ein Wort mit weniger Silben. Zum Beispiel hat das Wort „Schreibtisch “ zwei Silben aber 12 Buchstaben und ist damit deutlich länger als „Ironie“ mit 3 Silben.
Umgesetzt wurde die Berechnung der Silben der Zielwörter mithilfe der Pyphen-Bibliothek. Pyphen ist ein Modul zur Silbentrennung, welche dafür auf Hunspell Silbentrennungswörterbücher zugreift (pyphen, 2024).

4.1.3 Frequenz im DWDS

Wie oft ein Wort im alltäglichen Sprachgebrauch oder in Forum, Zeitungen, Nachrichten etc. vorkommt ist häufig ein Indiz dafür, ob es komplex ist oder nicht. Das DWDS-Wörterbuch (Digitales Wörterbuch der deutschen Sprache) umfasst Informationen zu Form und Bedeutung von über 230.000 Wörtern und wird fortlaufend aktualisiert. Es setzt sich aus vielen Textkorpora zusammen die vom Jahr 1897 bis heute reichen. Dazu gehören Zeitungskorpora wie die FAZ (Frankfurter Allgemeine Zeitung) und Die ZEIT, Webkorpora und darunter auch Blogs und Sozialkorpora wie der Wikepedia-Korpus, Politische Reden, Nachrichten und viele mehr. Außerdem bietet es APIs, um verschiedene Wortinformationen abzufragen. Die für diese Arbeit interessanteste API ist die API für die Worthäufigkeit (Frequenzbarometer). Die Worthäufigkeit kann für jegliche Wörter, die absolut mindestens fünf mal in folgenden Korpora vorkommen:

DWDS-Kernkorpus: Das DWDS-Kernkorpus ist ein Referenzkorpus der deutschen Sprache des 20. Jahrhunderts, das von 2000 bis 2003 an der Berlin-Brandenburgischen Akademie der Wissenschaften erstellt wurde. Es dient als Grundlage für das Digitale Wörterbuch der Deutschen Sprache (DWDS). Das Korpus ist balanciert. Es enthält eine ausgewogene Verteilung von Texten aus fünf Genres: Journalismus, Literatur, wissenschaftliche Texte, non-fiktionale Texte und gesprochene Sprache (Geyken, 2007).
Das DWDS-Kernkorpus hat das Ziel, die deutsche Sprache des 20. Jahrhunderts genau zu dokumentieren und zu analysieren. Es dient als Grundlage für Wörterbücher, Grammatiken und andere sprachliche Arbeiten. Im Gegensatz zu älteren Wörterbüchern, die meist nach dem Alphabet sortiert sind und auf der manuellen Auswahl einzelner Wörter und Sätze basieren, bietet das elektronische DWDS-Korpus genauere Informationen über die Häufigkeit von Wörtern und ihre Verbindungen. Es enthält wichtige literarische und wissenschaftliche Texte sowie Alltagsliteratur, um ein möglichst vollständiges Bild der deutschen Sprache in dieser Zeit zu zeigen.
Metakorpus WebXL: Das WebXL ist ein großes Webkorpus, das eine Vielzahl von deutschsprachigen Webseiten aus Ländern wie Deutschland, Österreich und der Schweiz umfasst. Es wurde 2020 im Portal des Digitalen Wörterbuchs der Deutschen Sprache (DWDS) zugänglich gemacht und besteht aus mehreren spezifischen Webkorpora, wie zum Beispiel dem Corona-Korpus, Jurakorpus, Medizinkorpus und verschiedenen Blogs (zum Beispiel IT- und Mode-Blogs). Das Korpus umfasst mehrere hunderttausend Webseiten, die sowohl aus professionellen (Nachrichten) als auch privaten Quellen (zum Bei-

spiel Vereine) stammen. Es zielt darauf ab, eine breite Vielfalt von Sprachverwendungen abzubilden.

DWDS-Zeitungskorpus: Das DWDS-Zeitungskorpus setzt sich aus vielen Tages- und Wochenzeitungen, verteilt über ganz Deutschland zusammen. Wie die FAZ und SPIEGEL Online gehört die BILD ebenfalls dazu.

Wikipedia-Korpus: Der Wikipedia-Korpus ist eine umfangreiche Sammlung von Texten, die aus den Inhalten der deutschsprachigen Wikipedia stammen.

Tabelle 4.1 zeigt, in welche Größenordnungen die Textkorpora einzuordnen sind.

Textkorpus	Sätze	Tokens	Stand
DWDS-Kernkorpus (1900-1999)	5.821.108	121.494.429	11.11.2021
Metakorpus WebXL (1995-2024)	1.775.947.139	25.176.891.646	28.05.2024
DWDS-Zeitungskorpus (1946-2024)	1.634.224.853	25.094.091.019	18.08.2024
Wikipedia-Korpus	60.802.237	1.278.443.617	22.07.2024

Tabelle 4.1: Größe der Textkorpora Digitales Wörterbuch der deutschen Sprache (DWDS) (2024)

Es gibt insgesamt sieben Häufigkeitsstufen für ein Wort. Wörter, die äußerst selten vorkommen (bis zu fünf Mal absolut), werden der Stufe 0 zugeordnet, während die am häufigsten vorkommenden Wörter die Stufe 6 erhalten. Es ist zu beachten, dass diese Stufen eine logarithmische Skala abbilden. Mehrwortausdrücke werden bei der Anfrage nicht berücksichtigt und erhalten im Datensatz den Wert 0.

Zum Zeitpunkt der Anfrage (10.04.2024) wurden für das angefragte Wort keine Flexionsformen berücksichtigt. Konjugierte oder deklinierte Wörter lieferten demnach eine andere Häufigkeitsstufe als das Lemma. So erhielt man zum Beispiel für das Wort „ausgewogene“ den Wert 0 und für das Lemma „ausgewogen“ den Wert 3. Aus diesem Grund wurde sich dazu entschieden, sowohl die Frequenz des Lemmas als auch die des Originalwortes abzufragen, in der Annahme, dass die Frequenz des Lemmas eine höhere Auswirkung auf die Klassifizierung hat. Die Lemmatisierung erfolgte mit dem Lemmatizer von Spacy[1]. Zu beachten ist, dass Wortsequenzen bei der Frequenzabfrage nicht berücksichtigt werden und den Wert 0 erhalten.

[1] https://spacy.io/api/lemmatizer, letzter Zugriff: 2024-10-02

4.1.4 Synonyme und Synsets

Für alle Zielwörter wurden sowohl die Anzahl an Synsets, in denen sie einen Platz haben, als auch ihre Synonyme erfasst. Unter einem Synonym versteht man in der Sprachwissenschaft ein Wort, welches zu einem anderen Wort von ähnlicher oder gleicher Bedeutung ist [2]. In einem Synset ist eine Gruppe von Wörtern vertreten, die in ihrer Semantik gleich sind. Man kann es also als eine Synonymgruppe von Wörtern betrachten. In dem verwendeten Datensatz werden diese beiden Merkmale ergänzt. Die Idee dahinter ist, dass Wörter, die viele verschiedene semantische Bedeutungen haben oder viele Synonyme besitzen, was beides oft zusammen einhergeht, häufiger als „nicht komplex“ klassifiziert werden, als welche, die weniger semantische Bedeutungen aufweisen, da Wörter mit vielen Bedeutungen häufiger Verwendung in der Sprache finden.
Um beide Werte zu erheben wurde OpenThesaurus verwendet. OpenThesaurus ist ein offenes deutsches Wortnetz, welches Synonymgruppen von Wörtern verfasst (Naber, 2005). Es bietet eine Web-API, über die man die benötigten Informationen anfragen und in XML- oder json-Format bekommt. Die Synsets wurden einfach zusammen addiert. Um die Anzahl der Synonyme zu bekommen, wurde die Anzahl von allen Synonymen innerhalb der Synsets addiert. Ausgenommen waren hier das Wort selbst und umgangssprachliche Synonyme.

4.1.5 Kosinus-Ähnlichkeit

Ein weiteres Merkmal für die Komplexität eines Wortes ist die Kosinus-Ähnlichkeit des Wortes zu allen anderen Wörtern im gleichen Satz. Ausgenommen sind dabei Stoppwörter sowie das Wort selbst. Die Stoppwörter wurden vor den Berechnungen entfernt, da sie in der Regel wenig semantische Bedeutung für den Satzinhalt haben und hauptsächlich der Strukturierung dienen. Zu den Stoppwörtern zählen beispielsweise Artikel, Präpositionen, Konjunktionen oder Hilfsverben wie „sein“ oder „haben“. Es wird angenommen, dass ein Wort durch die Berechnung der Kosinus-Ähnlichkeit in manchen Sätzen als komplex erscheint, in anderen hingegen weniger. Das kommt natürlich vor allem auf die Länge des Satzes an. Dieses Merkmal ist das einzige, das nicht nur das zu klassifizierende Wort selbst betrachtet, sondern auch den Kontext des gesamten Satzes berücksichtigt.

[2] ttps://www.dwds.de/wb/Synonym, letzter Zugriff: 2024-10-02

Um die Kosinus-Ähnlichkeit zwischen zwei Wörtern zu bestimmen, müssen zunächst beide Wörter in Word Embeddings umgewandelt werden. Dazu wird ein auf deutschen Wikipedia-Daten trainiertes Word2vec-Modell (Yamada u. a., 2020)[3] verwendet. Anschließend wird der Kosinus des Winkels zwischen den beiden Vektoren im mehrdimensionalen Raum berechnet, um ihre Ähnlichkeit zu ermitteln. Für zwei Vektoren A und B sieht das wie folgt aus:

$$\text{Kosinus-Ähnlichkeit} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \tag{4.1}$$

Um für jedes Zielwort einen numerischen Wert als Merkmal zu erhalten, mit dem das Modell arbeiten kann, wird der Mittelwert der für ein Wort berechneten Ähnlichkeitswerte verwendet. Im verwendeten Datensatz gibt es jedoch auch Mehrwortausdrücke. In diesen Fällen wird zunächst für jedes einzelne Wort ein Embedding-Vektor erzeugt. Aus denen wird dann ein neuer Vektor gebildet, der den Mittelwert der Einzelvektoren darstellt. Für diesen Vektor wird anschließend die Kosinus-Ähnlichkeit zu allen anderen Wörtern im Satz berechnet.

4.2 Random Forest Classifier

Unter den Teams, die ihre Systeme für den CWI Shared Task 2018 veröffentlicht haben, erzielten die besten Ergebnisse auf dem deutschen Datensatz diejenigen, die Random Forest Classifiers (RFC) einsetzten (Kajiwara und Komachi, 2018; Bingel und Bjerva, 2018). Ein RFC ist ein Algorithmus des maschinellen Lernens, der die Ergebnisse mehrerer Entscheidungsbäume kombiniert, um so eine präzisere Vorhersage zu erzielen. Durch das Bilden des Mittelwertes der Entscheidungen der Einzelbäume wird die Vorhersagegenauigkeit verbessert und die Gefahr des Overfittings verringert[4]. Die Systeme der TMU (Kajiwara und Komachi, 2018) belegten in vielen Klassifikationsaufgaben des CWI Shared Task 2018 mit die besten Plätze. Für die binäre Klassifikation verwendeten sie RFCs, die Merkmale wie Wortlänge und Häufigkeit innerhalb verschiedener Textkorpora umfassten. Auch das Department für Informatik der Universität Kopenhagen (CoastalCPH)

[3] https://huggingface.co/Word2vec/wikipedia2vec_dewiki_20180420_100d, letzter Zugriff: 2024-04-25

[4] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, letzter Zugriff: 2024-07-24

(Bingel und Bjerva, 2018) setzte für die binäre Klassifikation auf dem deutschen Datensatz RFCs ein und erreichte mit ihren Systemen eine Platzierung unter den Top 10. In dieser Arbeit wird ebenfalls ein RFC mit den zuvor beschriebenen Merkmalen verwendet. Der RFC dient hauptsächlich als Benchmark, an der sich die BERT-Modelle messen können. Zudem soll der RFC die Wichtigkeit der einzelnen Merkmale für die Klassifikation verdeutlichen und wertvolle Hinweise für die spätere Anwendung in den BERT-Modellen liefern.

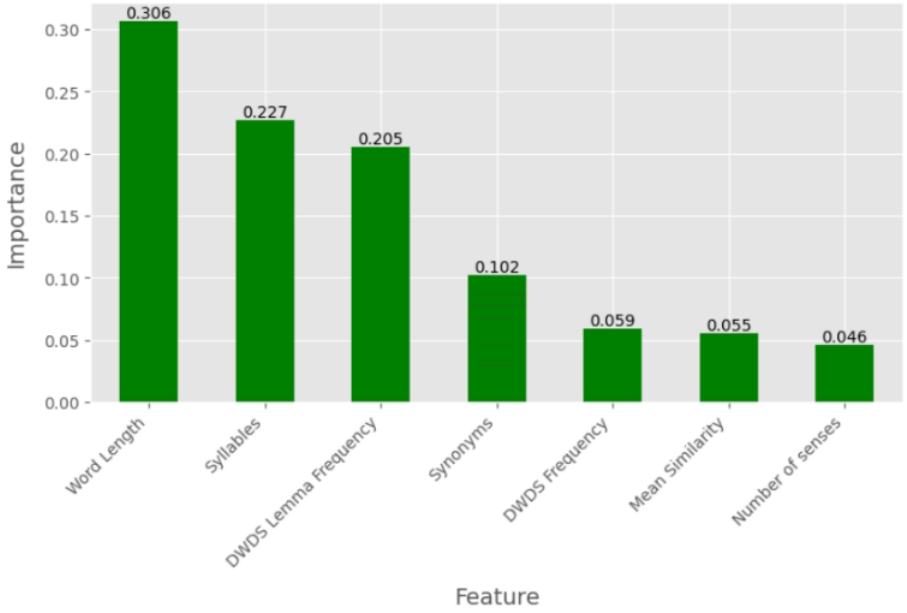


Abbildung 4.1: Wichtigkeit der Merkmale.

Zunächst wurde der RFC mit allen sieben Merkmalen trainiert. Um die besten Hyperparmeter zu finden, wurde das GridSearchCV-Modul[5] aus der sklearn-Bibliothek benutzt. So wurden sämtliche Kombinationen folgender Parameter getestet:

- n estimators: {10, 50, 100, 500, 1000}
- max depth: {5, 10, 15, 20, ∞}
- min samples leaf: {1, 5, 10, 15, 20}
- min samples split: {2, 5, 10, 20}

[5] https://scikit-learn.org/dev/modules/generated/sklearn.model_selection.GridSearchCV.html, letzter Zugriff: 2024-10-02

Zum großen Teil hatte Kajiwara und Komachi (2018) (Team TMU vom CWI 2018 Shared Task) Einfluss auf diese Parameterwahl. Ein Teil eines Entscheidungsbaums ist in Abbildung 1 im Anhang A.2 zu sehen. Da für die Evaluierung im CWI Shared Task 2018 die macro-averaged F1-Score als Metrik benutzt wurde, wird diese auch hier benutzt. In Abbildung 4.1 ist die Wichtigkeit der Merkmale zu sehen. Wie bereits vermutet, ist die Frequenz des Lemmas eines Wortes innerhalb des DWDS-Korpus ein deutlich besserer Indikator als die Frequenz des Wortes selbst. Nun wurde in der Wichtigkeit ansteigend ein Merkmal nach dem anderen entfernt und erneut trainiert. Zunächst wurden also die Anzahl der Bedeutungen (Number of senses) vernachlässigt, im nächsten Schritt zusätzlich auch die mittelwertgebildete Kosinus-Ähnlichkeit (Mean Similarity) usw. Bei einem Merkmal wird demnach nur noch die Wortlänge berücksichtigt. Die Ergebnisse sind in Tabelle 4.2 zu sehen.

Merkmale	F1-Score (macro)
7	0.7561
6	0.7569
5	0.7478
4	0.7458
3	**0.7617**
2	0.7228
1	0.7237

Tabelle 4.2: RFC Vergleich mit unterschiedlicher Anzahl an Merkmalen

Das beste Ergebnis wurde nur mit den 3 aussagekräftigsten Merkmalen erzielt, auch wenn der Unterschied zu den Ergebnissen mit mehr Merkmalen fast zu vernachlässigen ist. Es zeigt sich hier, dass ein RFC mit Merkmalen, welche sich auf die Wortlänge beziehen, für die Aufgabe gut geeignet ist, da mit den hier erzielten Ergebnissen die Baseline für die binäre Erkennung komplexer Wörter aus Yimam u. a. (2018) bereits überschritten wurde.

4.3 BERT

In diesem Kapitel wird das Ziel verfolgt, geeignete BERT-Modelle für die Aufgabe der CWI auszuwählen und diese durch verschiedene Optimierungstechniken wie Hyperparameter-Tuning, Datenaugmentation und die Einbeziehung der bereits besprochenen linguistischen Merkmale bestmöglich auf diese Aufgabe zuzuschneiden.

4.3.1 Modellwahl

Zunächst geht es darum, geeignete Modelle für die Aufgabe der CWI zu finden. Sämtliche in dieser Arbeit verwendeten Modelle sind bereits vortrainierte Modelle, die frei über den HuggingFace Model Hub[6] zugänglich sind. Der HuggingFace Model Hub ermöglicht es Mitgliedern der Community, ihre Model-Checkpoints zu speichern und mit anderen zu teilen. Bei der Auswahl der Modelle wurde darauf geachtet, dass es sich um Encoder-Only-Modelle handelt, da der bidirektionale Ansatz hier von großer Bedeutung ist, während kein Bedarf besteht, ganze Sätze oder Texte zu generieren. Der Auswahlprozess der Modelle vom HuggingFace Model Hub ist in Abbildung 4.2 visualisiert. Insgesamt bietet der Model Hub 1.000.003 Modelle an. Zunächst wurde nach der Aufgabe und der Sprache, auf der das Modell vortrainiert wurde, gefiltert. Die Aufgabe „Fill-Mask“ umfasst Modelle, die auf das Vorhersagen von fehlenden oder maskierten Wörtern in einem Satz optimiert sind. Durch die gezielte Suche nach Modellen mit Keywords wie „Bert“ oder „Distilbert“ und anhand von Benutzer-Feedback, wie der Anzahl der Downloads, blieben zehn geeignete Modelle übrig, von denen vier Modelle in die engere Auswahl aufgenommen wurden:

$\text{GBERT}_{\text{BASE}}$ & $\text{GBERT}_{\text{LARGE}}$:
$\text{GBERT}_{\text{BASE}}$ und $\text{GBERT}_{\text{LARGE}}$ (Chan u. a., 2020) sind zwei deutschsprachige BERT-Modelle, welche in Zusammenarbeit von den Entwicklern des originalen BERT (Devlin u. a., 2018) und DBMDZ BERT[7] trainiert wurden. Im Vergleich zu vorherigen BERT-Modellen wurde die Trainingsdatenmenge vergrößert sowie Whole Word Masking (WWM) angewandt. Da BERT eine WordPiece-Tokenisierung verwendet (Devlin u. a., 2018), werden oftmals nur Teilwörter maskiert. Beim WWM wird der Maskierungsansatz stattdessen so erweitert, dass immer ganze Wörter maskiert werden und nicht nur einzelne

[6] https://huggingface.co/models, letzter Zugriff: 2024-09-25
[7] https://github.com/dbmdz/berts, letzter Zugriff: 2024-09-25

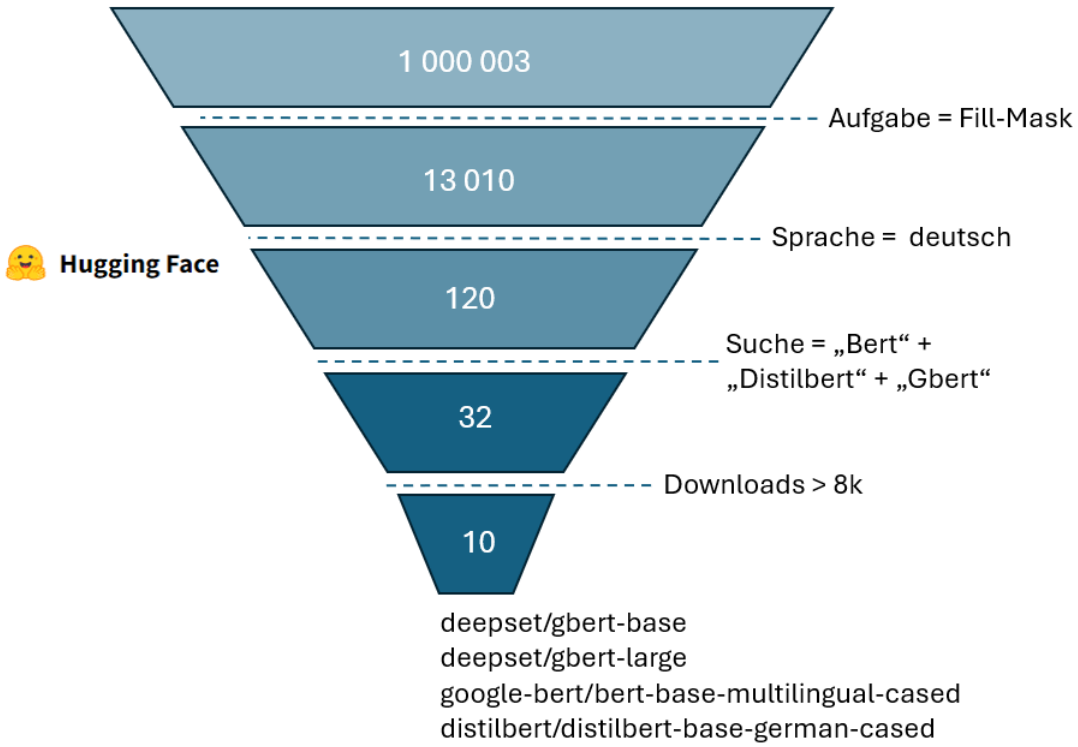


Abbildung 4.2: Auswahl der Modelle nach Kriterien (Stand: 2024-09-26)

Teilwörter oder Tokens. Das kann im Hinblick auf die Erkennung komplexer Wörter hilfreich sein. $\text{GBERT}_{\text{BASE}}$ und $\text{GBERT}_{\text{LARGE}}$ unterscheiden sich im Wesentlichen durch ihre Architektur. $\text{GBERT}_{\text{LARGE}}$ besitzt doppelt so viele Layer und mehr Hidden-States und Attention-Heads, sollte also in der Aufgabe der CWI hier etwas leistungsfähiger sein. Beide Modelle wurden zwar weniger oft heruntergeladen als DBMDZ BERT, haben aber in mehreren Aufgaben, darunter auch Klassifikationsaufgaben wie GemEval18 (Wiegand u. a., 2018) bessere Leistungen erzielt und wurden deshalb bevorzugt.

DistilBERT:
DistilBERT ist eine komprimierte und effizientere Version des BERT-Modells (Devlin u. a., 2018). Es ist 40 % kleiner und arbeitet 60 % schneller, während es 97 % der Sprachverständnisfähigkeiten von BERT beibehält (Sanh, 2019). Diese Effizienzsteigerung wird durch den Prozess der Knowledge Distillation erreicht (Buciluǎ u. a., 2006; Hinton, 2015). Knowledge Distillation ist eine Technik der Modellkompression, die erstmals von Buciluǎ u. a. (2006) vorgestellt wurde. Dabei wird das Wissen eines großen,

komplexen Modells (dem sogenannten „Teacher-Modell“) auf ein kleineres, ressourceneffizienteres Modell (das „Student-Modell“) übertragen, ohne dass es zu einem signifikanten Leistungsverlust kommt. Dies ermöglicht es, leistungsstarke Modelle mit deutlich geringerem Ressourcenaufwand einem breiteren Nutzerkreis zugänglich zu machen.

Im Rahmen der Knowledge Distillation wird zunächst das Teacher-Modell trainiert, um sogenannte “ Soft Targets“ zu erzeugen. Diese stellen Wahrscheinlichkeitsverteilungen der Klassen dar, die mehr Informationen enthalten als nur das korrekte Label. Das Student-Modell wird anschließend mit diesen Soft Targets sowie den echten Labels trainiert. Ein wesentlicher Vorteil dieses Verfahrens besteht darin, dass das Student-Modell so nicht nur lernt, welche Klasse am wahrscheinlichsten ist, sondern auch, wie sicher oder unsicher das Teacher-Modell in seinen Vorhersagen war. Dadurch kann das Student-Modell eine bessere Generalisierungsfähigkeit entwickeln.

Das verwendete DistilBERT-Modell wurde auf der Hälfte der Daten des DBMDZ-BERT über einen Zeitraum von etwa vier Tagen trainiert und zeigte bei Named Entity Recognition (NER)-Aufgaben nur minimale Leistungsunterschiede [8]. DistilBERT wurde für diese Arbeit ausgewählt, um zu untersuchen, wie gut sich ein kleineres Modell im Vergleich zu größeren Modellen bei der Lösung der CWI-Aufgabe behaupten kann.

BERT$_{\text{MULTILINGUAL}}$: Dieses Modell basiert auf dem originalen BERT (Devlin u. a., 2018) und wurde auf den 104 Sprachen mit den größten Wikipedia-Einträgen trainiert. Da der deutsche Datensatz des CWI 2018 Shared Task hauptsächlich aus Wikipedia-Einträgen besteht, könnte der Einsatz eines multilingualen Modells vorteilhaft sein, insbesondere bei der Erkennung von Anglizismen und Fremdwörtern aus anderen Sprachen. Die sprachübergreifende Generalisierungsfähigkeit könnte zu guten Leistungen beitragen.

Eine genauere Übersicht über die Parameter der ausgewählten Modelle sowie die Daten, auf denen sie vortrainiert wurden, sind in Tabelle 4.3 aufgelistet.

Das Finetuning großer Transformer-Sprachmodelle auf einem vergleichsweise kleinen Datensatz kann zu bekannten Problemen wie Overfitting führen (Xu u. a., 2020). Overfitting ist ein bekanntes Phänomen im maschinellen Lernen, bei dem ein Modell so stark auf die Trainingsdaten angepasst wird, dass es Schwierigkeiten hat, auf neue, unbekannte Daten gut zu generalisieren (Ying, 2019). Dies liegt daran, dass das Modell nicht nur

[8] https://github.com/huggingface/transformers/pull/1873, letzter zugriff: 2024-09-2024

Modell	Parameter	Pre-Training Datensatz	Paper
$\text{GBERT}_{\text{BASE}}$	111 Mio	OSCAR(Suárez u. a., 2019), OPUS, Wikipedia, OpenLegalData	Chan u. a. (2020)
$\text{GBERT}_{\text{LARGE}}$	337 Mio	OSCAR(Suárez u. a., 2019), OPUS, Wikipedia, OpenLegalData	Chan u. a. (2020)
DistilBERT	67.4 Mio	Wikipedia dump, EU Bookshop corpus, Open Subtitles, CommonCrawl, ParaCrawl, News Crawl	Sanh (2019)
$\text{BERT}_{\text{MULTILINGUAL}}$	179 Mio	Wikipedia	Devlin u. a. (2018)

Tabelle 4.3: Benutzte BERT-Modelle. (Jeweils letztes Zugriffsdatum 26.09.2024)

die zugrunde liegenden Muster in den Daten lernt, sondern auch das Rauschen oder zufällige Unregelmäßigkeiten in den Trainingsdaten. Anstatt allgemeine Regeln zu lernen, die auch auf neue Daten anwendbar sind, memoriert das Modell die Trainingsdaten und generalisiert somit schlechter. Es gibt jedoch verschiedene Methoden, die dem Overfitting entgegenwirken können:

- *Frühzeitiges Stoppen (Early Stopping)*: Das Training wird beendet, bevor das Modell beginnt, die Trainingsdaten auswendig zu lernen. Dies ist oft daran erkennbar, dass der Validierungsloss steigt, während der Trainingsloss weiter sinkt.
- *Komplexitätsreduktion*: Durch die Reduzierung der Modellkomplexität wird verhindert, dass das Modell unnötige Details und Rauschen aus den Trainingsdaten lernt.
- *Datenerweiterung (Data Augmentation)*: Bei komplexen Modellen kann das Hinzufügen zusätzlicher Daten helfen, die Generalisierungsfähigkeit des Modells zu verbessern und Overfitting zu vermeiden.
- *Regularisierung*: Techniken wie das Bestrafen zu großer Gewichte (L1- oder L2-Regularisierung) reduzieren den Einfluss weniger relevanter Merkmale und fokussieren das Modell auf wichtigere Informationen.
- *Dropout*: Beim Dropout wird während des Trainings zufällig ein Teil der Neuronen deaktiviert, sodass das Modell gezwungen wird, robuste Merkmalsrepräsentationen zu lernen, die nicht nur auf einzelnen Neuronen basieren.

Alle vier Modelle wurden auf die Aufgabe der CWI und den deutschen Datensatz des CWI 2018 Shared Task hin feinabgestimmt. Hierfür wurde dem vortrainierten Modell eine Klassifikationsschicht speziell für Sequenzen hinzugefügt, da nicht nur einzelne Tokens, sondern teilweise mehrere Wörter klassifiziert werden müssen. Um Overfitting zu

vermeiden, wurde zwischen dem BERT-Embedding-Layer und der Klassifikationsschicht ein Dropout-Layer mit einer Rate von 0,1 eingefügt. Zudem wurde ein Early-Stopping Callback aus der Transformer-Bibliothek[9] implementiert, das während des Trainingsprozesses aufgerufen wurde. Dabei wurde Patience von 4 angewendet, was bedeutet, dass das Training beendet wird, sobald sich der Validierungsloss in vier aufeinanderfolgenden Evaluierungen nicht verbessert. Die Hyperparameter für $\text{GBERT}_{\text{BASE}}$ und $\text{GBERT}_{\text{LARGE}}$ wurden der zugehörigen Publikation von Chan u. a. (2020) entnommen. Das Training erfolgte mit einer Batch-Größe von 32, einer Lernrate von 5e-06 und einer Epochenanzahl von 5. Diese Hyperparameter wurden auch für die Feinabstimmung verwendet. $\text{BERT}_{\text{MULTILINGUAL}}$ wurde mit einer Batch-Größe von 16, einer Lernrate von 4e-05 und einer Epochenanzahl von 4 trainiert, was gemäß Devlin u. a. (2018) als geeignete Werte gilt. Für DistilBERT wurden die gleichen Hyperparameter wie für $\text{GBERT}_{\text{BASE}}$ und $\text{GBERT}_{\text{LARGE}}$ verwendet. Das Training wurde für die vier Modelle jeweils fünfmal mit unterschiedlichen Seeds durchgeführt, um aussagekräftigere Ergebnisse zu erhalten. Anschließend wurde jeweils der Medianwert anstelle des Mittelwerts aller Testdurchläufe verwendet, um weniger anfällig gegenüber Ausreißern zu sein. Die Ergebnisse für alle vier Modelle sind in Tabelle 4.4 aufgeführt.

Modell	F1-Score
$\text{GBERT}_{\text{BASE}}$	0.7744
$\text{GBERT}_{\text{LARGE}}$	**0.7794**
DistilBERT	0.7449
$\text{BERT}_{\text{Multilingual}}$	0.7590

Tabelle 4.4: Ergebnisse der BERT-Baseline Modelle bei der binären Klassifikation auf dem Datensatz des CWI Shared Task 2018

Aus Tabelle 4.4 geht hervor, dass beide GBERT-Modelle die Baseline von 0.7546 aus dem CWI 2018 Shared Task übertroffen haben. Die Annahme, dass größere Modelle in der Regel bessere Ergebnisse liefern, hat sich auch in diesem Fall bestätigt, da DistilBERT mit Abstand die schlechtesten Ergebnisse erzielte. Für die weiteren Optimierungen wird $\text{GBERT}_{\text{BASE}}$ als Baseline verwendet. Obwohl $\text{GBERT}_{\text{LARGE}}$ minimal bessere Ergebnisse erreicht hat, rechtfertigt dies den erhöhten Ressourcenaufwand für die weiteren Experimente nicht. Um eine aussagekräftigere Baseline auf den Originaldaten zu erhalten, wird im nächsten Kapitel zunächst eine Hyperparameter-Optimierung durchgeführt. Anschlie-

[9] https://huggingface.co/docs/transformers/main_classes/callback, letzterZugriff:2024-10-03

ßend erfolgt die Erweiterung des Datensatzes sowie die Ergänzung von Merkmalen, die bereits im RFC verwendet wurden.

4.3.2 Hyperparameter Optimierung

Die Hyperparameter Optimierung wird automatisiert. Folgende Hyperparameter werden bei der Suche berücksichtigt:

Learning Rate (Lernrate):
Die Lernrate ist einer der wichtigsten Hyperparameter, der den Schritt bestimmt, mit dem das Modell die Gewichte während des Trainings aktualisiert (Yang und Shami, 2020). Anders gesagt bestimmt sie Schrittgröße, mit der das Modell in Richtung des Minimums der Verlustfunktion bewegt wird. Eine hohe Lernrate kann das Modell beim Lernprozess beschleunigen, bringt jedoch auch das Risiko, dass das Minimum der Verlustfunktion überschritten wird, was dazu führen kann, dass das Modell nicht die optimale Lösung findet, sondern nur eine Annäherung in der Umgebung des Minimums erreicht. Ist die Lernrate zu groß, kann der Optimierer das Minimum sogar komplett überschießen und das Modell in einen Zustand divergenter Werte führen, was das Training instabil macht. Eine niedrige Lernrate führt dazu, dass das Modell langsamer lernt, aber der Lernprozess stabiler ist. Niedrige Lernraten bieten eine bessere Chance, das lokale Minimum zu erreichen und eine optimierte Verlustfunktion zu finden. Der Nachteil besteht darin, dass das Training mehr Epochen und damit auch mehr Zeit benötigt, um zu konvergieren.

Batch-Size (Batch Größe): Die Batchgröße beschreibt die Anzahl der Trainingsbeispiele, die das Modell in einem einzelnen Schritt verarbeitet, bevor die Gewichte aktualisiert werden. Eine sehr große Batchgröße hingegen führt zu stabileren und präziseren Gradientenaktualisierungen, da die Fehlerberechnung auf einer größeren Menge von Daten basiert. Der Nachteil großer Batches ist jedoch, dass sie mehr Speicher benötigen, was bei beschränkten GPU-Ressourcen problematisch sein kann.
Eine kleinere Batch-Größe führt hingegen zu häufigeren, aber ungenaueren Aktualisierungen der Modellgewichte. Dies kann manchmal von Vorteil sein, da die häufigeren Updates dazu beitragen können, dass das Modell schneller konvergiert und sich von lokalen Minima entfernt, in denen es sonst möglicherweise stecken bleibt. Während des Trainingsprozesses kann es aber zu stärkeren Schwankungen kommen.

Epochenanzahl: Die Anzahl der Epochen beschreibt, wie oft der gesamte Trainingsdatensatz während des Trainingsprozesses vom Modell durchlaufen wird. Dies bedeutet,

dass jede Trainingseinheit (Sample) wiederholt betrachtet wird, um die Modellparameter zu optimieren. Eine zu niedrige Anzahl an Epochen kann bei größeren Modellen häufig zu Underfitting führen, während eine zu hohe Anzahl an Epochen das Risiko von Overfitting erhöht.

Die Suche nach den optimalen Hyperparametern wird durch das Optuna-Framework automatisiert[10]. Zunächst wurde der Suchraum wie folgt definiert:

- learning_rate: {3e-06, 5e-06, 1e-05, 2e-05, 3e-05}
- batch_size: {8, 16, 32}
- num_eopchs: {3, 5, 7}
- weight_decay: {0.0, 0.01, 0.001}

Da die Lernrate mit einem Wert von $5e-06$ bereits als sehr niedrig ist, wurden überwiegend größere Werte untersucht. Die Batchgröße ist mit 32 im Verhältnis zur Größe des Datensatzes bereits relativ hoch; daher wurden zwei niedrigere Werte getestet. Da das Training der Baseline-Modelle teilweise durch Early-Stopping beendet wurde und der Validierungsloss nach kurzer Zeit stagnierte, werden keine hohen Werte für die Epochenzahl in Betracht gezogen. Zusätzlich wird eine Regularisierungstechnik in Form von Weight Decay getestet, bei der übermäßigen Gewichten während des Trainings eine „Strafe“ hinzugefügt wird, um potenziellem Overfitting entgegenzuwirken. Die Suche wird fünfmal wiederholt, mit jeweils unterschiedlichen Seeds, um die bestmögliche Kombination zu ermitteln. Die Zielmetrik, nach der das beste Modell zu bewerten ist, ist der macro-averaged F1-Score wie auch im CWI 2018 Shared Task.

4.3.3 Datenaugmentierung

Unabhängig von der spezifischen Aufgabe im Bereich der Verarbeitung natürlicher Sprache (NLP) hängt die Leistung eines Modells maßgeblich von der Qualität und der Menge der verfügbaren Daten ab. Da die Datenerfassung und -annotation im NLP-Bereich oft mit erheblichen Herausforderungen verbunden ist und einen hohen zeitlichen Aufwand erfordert, sind Methoden zur Datenaugmentierung (DA) von zentraler Bedeutung. DA bezeichnet Strategien, die darauf abzielen, die Vielfalt der Trainingsdaten zu erhöhen,

[10] https://optuna.org/, letzter Zugriff: 2024-10-01

ohne dass zusätzliche Daten akquiriert werden müssen (Feng u. a., 2021). Diese Techniken können die Modellleistung verbessern, ohne dass Änderungen an der Modellarchitektur notwendig sind. Allerdings ist der Einsatz von DA im NLP im Vergleich zur Computer Vision (CV) noch weniger erforscht, was größtenteils auf die hohe Komplexität natürlicher Sprache zurückzuführen ist. Dennoch existieren einige bewährte Ansätze.

Die von Wei und Zou (2019) vorgestellten Easy Data Augmentation (EDA)-Techniken umfassen hauptsächlich tokenbasierte Verfahren, die auf einfache Weise Variationen in den Trainingsdaten erzeugen. Zu diesen Methoden gehören:

- **Random Swap** (Zufälliges Vertauschen): Zwei zufällig ausgewählte Wörter im Satz werden vertauscht, um die Reihenfolge zu ändern, ohne den Gesamtkontext oder die Bedeutung erheblich zu beeinflussen.
- **Random Deletion** (Zufälliges Entfernen): Ein oder mehrere zufällige Wörter werden aus dem Satz gelöscht. Dadurch lernt das Modell, auch mit unvollständigen oder unklaren Informationen umzugehen.
- **Random Insertion** (Zufälliges Einfügen): Ein zufällig ausgewähltes Wort wird in den Satz eingefügt, wobei darauf geachtet wird, dass das Wort semantisch oder thematisch in den Kontext passt.
- **Synonym Replacement** (Synonym-Ersetzung): Eines oder mehrere Wörter im Satz werden durch ihre Synonyme ersetzt. Beispielsweise könnte das Wort „groß“ durch „riesig“ ersetzt werden. Diese Methode erfordert jedoch eine sorgfältige Auswahl der Synonyme, um sicherzustellen, dass die Bedeutung des Satzes erhalten bleibt und keine semantischen Veränderungen auftreten.

Obwohl diese EDA-Techniken zur Diversifizierung von Trainingsdaten beitragen, besteht stets das Risiko, dass der Satzkontext verändert wird. Dieses Risiko variiert insbesondere in Abhängigkeit von der Länge und Komplexität des Satzes.

Eine bewährte Methode zur Datenaugmentierung, die auf Satzebene angewendet wird, ist die **Backtranslation** (Rückübersetzung) (Sennrich u. a., 2016). Bei diesem Ansatz wird ein Satz zunächst in eine andere Sprache übersetzt und anschließend in die Ausgangssprache zurückübersetzt. Ziel ist es, natürliche Variationen im Satzaufbau und der Wortwahl zu erzeugen, während die ursprüngliche Bedeutung erhalten bleibt.

In dieser Arbeit liegt der Schwerpunkt auf der Methode der Backtranslation, die unter Verwendung von MarianMT (Junczys-Dowmunt u. a., 2018) implementiert wurde, einem Framework für Neural Machine Translation (NMT), das Übersetzungsmodelle für zahlreiche Sprachen bereitstellt.

Eine besondere Herausforderung bei der Backtranslation im Kontext der CWI besteht darin, dass das Zielwort im übersetzten Satz unverändert erhalten bleiben muss. Um dies zu gewährleisten, wird das Zielwort vor der Übersetzung in die Zielsprache durch ein `[TARGET]`-Token als Platzhalter ersetzt. Anschließend wird der gesamte Satz, mit Ausnahme des Zielwortes, in die Zielsprache übersetzt und anschließend wieder ins Deutsche zurückübersetzt. In einem letzten Schritt wird das `[TARGET]`-Token durch das ursprüngliche Zielwort ersetzt. Für dieses Verfahren werden zwei Übersetzungsmodelle verwendet: ein Modell für die Übersetzung von Deutsch nach Englisch und ein weiteres für die Übersetzung von Englisch nach Deutsch.

Diese Methode wird auf jedes Sample der Trainingsdaten angewendet, wodurch die Datenmenge verdoppelt wird. Auf diesem erweiterten Datensatz wird $\text{GBERT}_{\text{BASE}}$ mit den im vorherigen Kapitel ermittelten Hyperparametern fünf Durchläufe trainiert und daraus der Medianwert ermittelt.

4.3.4 Integration linguistischer Merkmale

In diesem Kapitel wird die Erweiterung der BERT-Embeddings um numerische Merkmale beschrieben, die auch im RFC verwendet wurden. Um es BERT zu ermöglichen, numerische Features zu verarbeiten und diese mit den erzeugten Textrepräsentationen zu kombinieren, wurde eine angepasste BERT-Klasse implementiert. Diese Klasse erwartet bei der Initialisierung die Anzahl der numerischen Merkmale. In Abbildung 4.3 ist die entsprechende Modellarchitektur dargestellt. Der Eingabetext wird durch ein `[CLS]`-Token eingeleitet, welches den Beginn des Satzes repräsentiert, sowie durch zwei `[SEP]`-Tokens ergänzt. Diese `[SEP]`-Tokens werden von BERT normalerweise verwendet, um zwei Sequenzen zu trennen; in diesem Fall dienen sie dazu, das Zielwort vom restlichen Eingabetext zu trennen. Nach der Tokenisierung durchläuft der Text das BERT-Modell, wobei die resultierenden Textrepräsentationen (BERT-Embeddings oder „pooled output “) erzeugt werden. Anschließend werden diese Embeddings mit den numerischen Merkmalen konkateniert, bevor der Klassifikationsschicht übergeben werden.

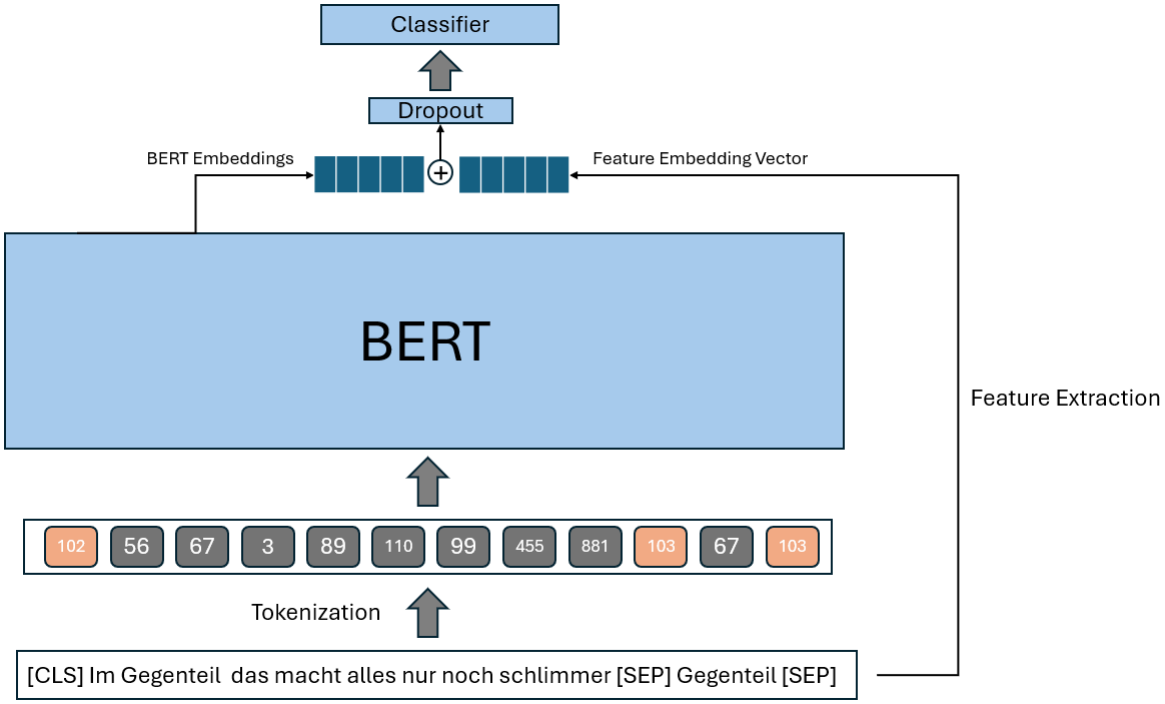


Abbildung 4.3: Modellarchitektur mit Einbeziehen von linguistischen Merkmalen

Das Modell wird ebenfalls mit den Hyperparametern trainiert, die durch die Hyperparameter-Optimierung ermittelt wurden. Der Trainingsprozess erfolgt in zwei Durchläufen:

1. *Drei Merkmale*: Training mit den drei relevantesten Merkmalen gemäß Abbildung 4.1.

2. *Sieben Merkmale*: Training mit allen im RFC verwendeten Merkmalen.

Die Überlegung dahinter ist, dass eine Vielzahl an Merkmalen das Modell möglicherweise unnötig komplex macht und die Leistung dadurch sogar verschlechtert werden könnte. Da sich die Bedeutung der drei wichtigsten Merkmale deutlich von den übrigen unterscheidet, wurde diese Aufteilung vorgenommen. Es besteht die Möglichkeit, dass diese drei Merkmale bereits ausreichen, um dem Modell eine gute Vorhersagegenauigkeit zu ermöglichen. Gleichzeitig könnten alle sieben Merkmale zusammen zu einer „Verwirrung“ des Modells führen und zusätzliches Rauschen erzeugen, das die Leistung beeinträchtigt. Beide Durchläufe werden wieder fünfmal mit verschiedenen seeds durchlaufen und der Medianwert ermittelt.

5 Ergebnisse & Analyse

In diesem Kapitel werden die Ergebnisse der in Kapitel 4 beschriebenen Methoden präsentiert und diskutiert. Die gezeigten Grafiken und Auswertungstabellen basieren in der Regel auf dem Median des macro-averaged F1-Scores aus insgesamt fünf Durchläufen, da dies auch die zentrale Metrik für die binäre Klassifikation im CWI Shared Task 2018 ist.

5.1 Ergebnisse und Analyse der Baseline-Modelle

Abbildung 5.1 zeigt den Verlauf des Trainings- und Validationloss, während Abbildung 5.2 die Konfusionsmatrix für die Vorhersagen auf dem Testdatensatz der vier Baseline-Modelle darstellt. Die detaillierten Metriken, einschließlich macro-averaged F1-Score, Precision, Recall und Accuracy, sind in Tabelle 5.1 aufgeführt. In Abbildung 5.1 ist zu erkennen, dass der Validationloss bei $\text{GBERT}_{\text{BASE}}$ und DistilBERT stetig sinkt und nahe am Trainingsloss liegt, was auf eine stabile Generalisierungsleistung dieser Modelle hinweist. Beide Modelle zeigen während des gesamten Trainings keine Anzeichen von Overfitting und trainieren die vollen fünf Epochen durch. Im Fall des größeren Modells $\text{GBERT}_{\text{LARGE}}$ sinkt der Trainingsloss ab Schritt 600 stark, während der Validationloss bei etwa 0,4 stagniert. Dies könnte darauf hindeuten, dass das Modell beginnt, sich zu stark an die Trainingsdaten anzupassen (Overfitting). Deshalb wurde das Training durch Early Stopping nach ca. 2,5 Epochen vorzeitig beendet, um weiteres Overfitting zu vermeiden.
Das $\text{BERT}_{\text{MULTILINGUAL}}$-Modell zeigt im Vergleich zu den anderen Modellen stärkere Schwankungen im Validationloss. Ab Schritt 800 wird ein klares Anzeichen von Overfitting sichtbar, da der Validationloss deutlich ansteigt, während der Trainingsloss weiter sinkt. Auch hier wurde das Training frühzeitig beendet, um eine weitere Überanpassung zu verhindern.

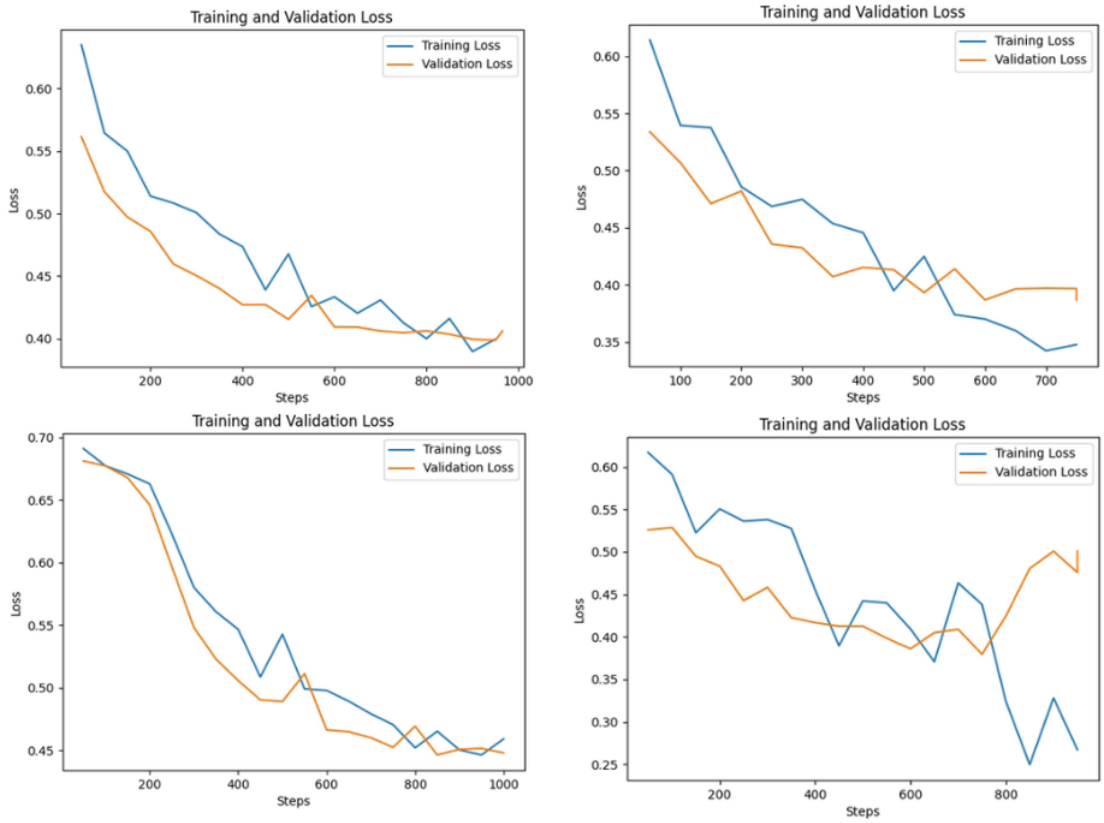


Abbildung 5.1: Trainings- und Validationloss der Baseline-Modelle $\text{GBERT}_{\text{BASE}}$, $\text{GBERT}_{\text{LARGE}}$, DistilBERT, $\text{BERT}_{\text{Multilingual}}$ (reihenweise von oben links nach unten rechts)

Abbildung 5.2 illustriert die Leistung der Modelle auf dem Testdatensatz. Hier wird deutlich, dass die drei deutschsprachigen Modelle $\text{GBERT}_{\text{BASE}}$, $\text{GBERT}_{\text{LARGE}}$ und DistilBERT nicht-komplexe Wörter zuverlässig erkennen, mit einer Genauigkeit von etwa 85 %. Bei der Klassifikation komplexer Wörter liegt die Genauigkeit jedoch niedriger: $\text{GBERT}_{\text{BASE}}$ und $\text{GBERT}_{\text{LARGE}}$ klassifizieren komplexe Wörter in etwa 69 % der Fälle korrekt, während das kleinere DistilBERT eine etwas geringere Genauigkeit von 62 % aufweist.

Interessanterweise verhält sich $\text{BERT}_{\text{MULTILINGUAL}}$ anders als die Modelle, die nur auf deutschen Daten vortrainiert wurden. Es klassifiziert komplexe Wörter mit einer höheren Genauigkeit von 78 %, während es nicht-komplexe Wörter nur zu 76 % korrekt identifiziert. Dies könnte die Annahme bestätigen, dass aufgrund der mehrsprachigen Trainingsdaten die Generalisierungsfähigkeit für die Klassifikation komplexer Wörter besser ist.

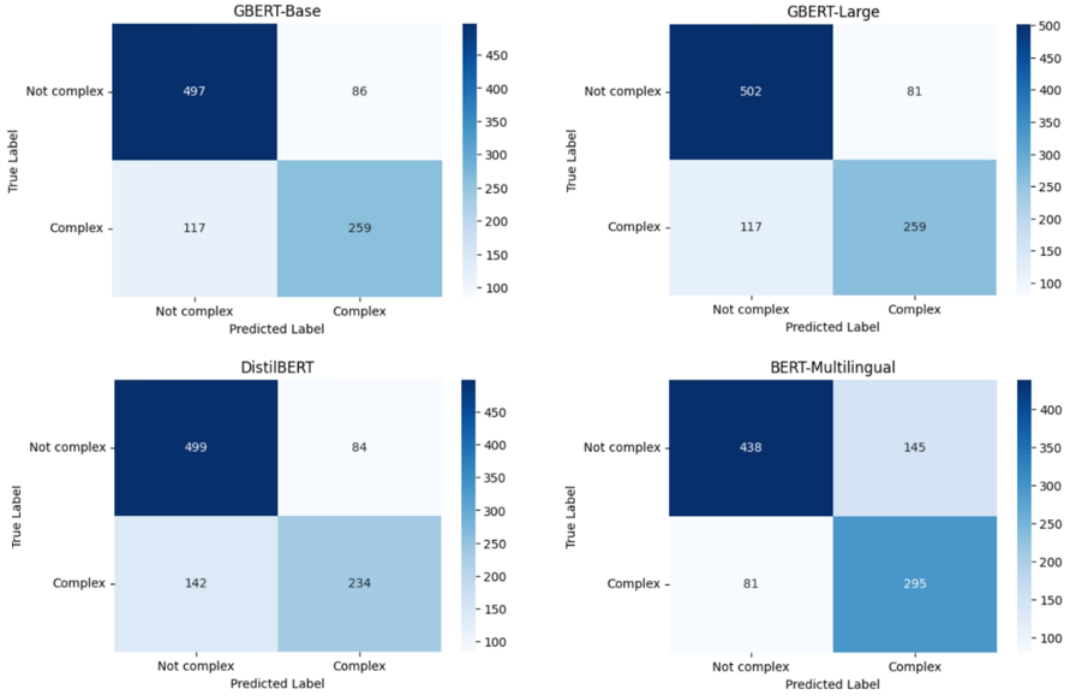


Abbildung 5.2: Konfusionsmatrizen der Baseline-Modelle $\text{GBERT}_{\text{BASE}}$, $\text{GBERT}_{\text{LARGE}}$, DistilBERT, $\text{BERT}_{\text{Multilingual}}$ (reihenweise von oben links nach unten rechts)

Zusammenfassend zeigen alle vier Modelle einen kontinuierlichen Abfall des Trainingsloss, was darauf hindeutet, dass sie erfolgreich die Trainingsdaten modellieren können. Der Validationloss ist jedoch bei einigen Modellen instabiler, was angesichts der Größe des Datensatzes, der mit 9.000 Samples für große Modelle wie BERT als eher klein gilt, und der Komplexität der Modelle zu erwarten ist. $\text{GBERT}_{\text{BASE}}$ liefert nahezu die gleichen Ergebnisse wie $\text{GBERT}_{\text{LARGE}}$, obwohl Letzteres dreimal so groß ist. Dies deutet darauf hin, dass für diesen Datensatz keine sehr komplexen Modelle mit deutlich höherem Ressourcenaufwand erforderlich sind.
DistilBERT zeigt ebenfalls eine gute Leistung bei der Erkennung nicht-komplexer Wörter, hat jedoch Schwierigkeiten bei der Klassifikation komplexer Wörter. Dies könnte auf die geringere Modellgröße und Kapazität zurückzuführen sein. $\text{BERT}_{\text{MULTILINGUAL}}$ liefert die beste Leistung bei der Identifikation komplexer Wörter, was wahrscheinlich auf die sprachübergreifende Generalisierungsfähigkeit des Modells zurückzuführen ist. Allerdings zeigt es eine höhere Fehlerrate bei der Klassifikation nicht-komplexer Wörter, was daran liegen könnte, dass es ausschließlich auf dem Wikipedia Korpus trainiert wurde.

Modell	F1-Score (macro)	Präzision	Recall	Accuracy
$\text{GBERT}_{\text{BASE}}$	0.77	0.78	0.76	0.85
$\text{GBERT}_{\text{LARGE}}$	**0.78**	**0.79**	0.76	**0.86**
DistilBERT	0.74	0.75	0.73	0.83
$\text{BERT}_{\text{MULTILINGUAL}}$	0.76	0.74	**0.78**	0.81

Tabelle 5.1: Metriken der Baseline-Modelle

Die in Tabelle 5.1 aufgeführten Metriken bestätigen, dass alle Modelle ohne Optimierungen wie das Hinzuziehen von linguistischen Merkmalen insgesamt eine solide Leistung erbringen, was aufgrund ihres Pretrainings auf großen Textkorpora und ihres bidirektionalen Ansatzes zu erwarten war. Auch bezogen auf die Größe der Modelle, schneidet $\text{GBERT}_{\text{BASE}}$ insgesamt am besten ab.

5.2 Ergebnisse und Analyse der Hyperparameter-Optimierung

Durch die Hyperparameter-Optimierung mit Optuna[1] wurden folgende Änderungen vorgenommen:

- *Ursprüngliche Konfiguration*: Lernrate: 5e-06, Batchgröße: 32, Epochen: 5
- *Optimierte Konfiguration*: Lernrate: 2e-05, Batchgröße: 32, Epochen: 7

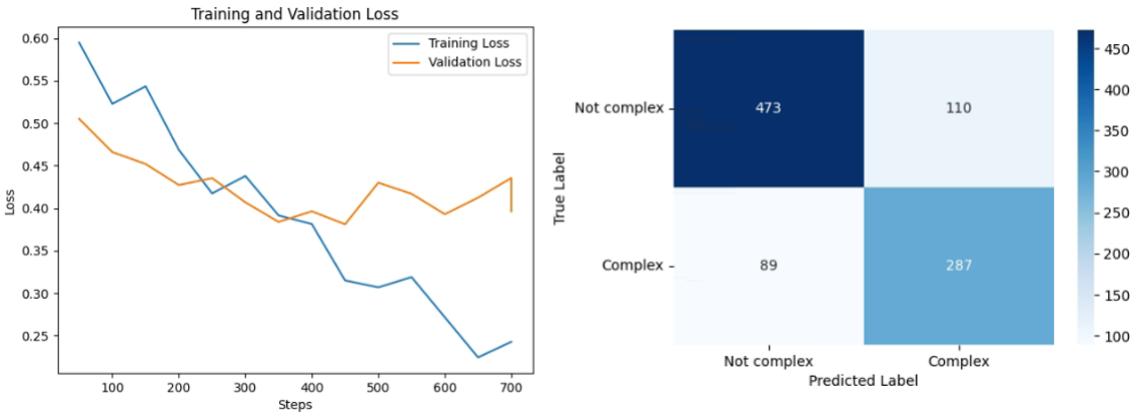


Abbildung 5.3: $\text{GBERT}_{\text{BASE}}$ - Ergebnisse nach Hyperparameteroptimierung

[1] https://optuna.org/, letzter Zugriff: 2024-10-02

Abbildung 5.3 präsentiert die Ergebnisse des $\text{GBERT}_{\text{BASE}}$ Modells nach der Optimierung der Hyperparameter auf dem Testdatensatz des CWI 2018 Shared Task. Trotz der Konfiguration für 7 Epochen wurde das Training des GBERT-Base-Modells nach 3,6 Epochen durch Early Stopping abgebrochen. Dies geschah, weil der Validationloss stagnierte und die Gefahr des Overfittings bestand. Ähnlich wie beim GBERT-Large-Modell zeigte sich, dass der Validationloss nach einer bestimmten Anzahl von Schritten bei etwa 0,4 stagnierte, während der Trainingsloss weiter sank. Dies ist ein typischer Indikator dafür, dass sich das Modell anfängt, die Trainigsdaten zu merken und die Generalisierungsfähigkeit sinkt. Durch die Wahl einer höheren Lernrate konnte das Modell jedoch schneller ein Minimum der Verlustfunktion erreichen. Während beim ursprünglichen GBERT-Base dieses Minimum erst nach etwa 800 Schritten erreicht wurde, gelang es dem optimierten Modell bereits nach 350 Schritten, ein stabileres Lernverhalten zu zeigen. Besonders positiv hervorzuheben ist die Verbesserung bei der Klassifikation komplexer Wörter. Die Erkennungsrate stieg von 69 % auf 76 %. Allerdings ging dies mit einer leichten Verschlechterung in der Klassifikation von nicht-komplexen Wörtern einher, deren Erkennungsrate von 85 % auf 81 % sank. Insgesamt hat die Hyperparameter-Optimierung Verbesserungen in allen wichtigen Metriken gebracht, wie Tabelle 5.2 demonstriert. Besonders hervorzuheben ist die Steigerung des macro-averaged F1-Scores.

Modell	**F1-Score (macro)**	**Precision**	**Recall**	**Accuracy**
$\text{GBERT}_{\text{BASE}}$	0.7744	0.7801	0.7707	0.7883
$\text{GBERT}_{\text{BASE}}$ (optimiert)	**0.7848**	**0.7823**	**0.7873**	**0.7925**

Tabelle 5.2: Vergleich der Metriken für $\text{GBERT}_{\text{BASE}}$ vor und nach der Hyperparameteroptimierung

5.3 Ergebnisse und Analyse der Datenaugmentierung

Bei der Datenaugmentierung wurde ausschließlich Backtranslation mit Englisch verwendet. In Abbildung 5.4 zeigt sich, dass die augmentierten Daten bereits nach ca. einer Epoche zu einem Overfitting führten und das Training durch Early-Stoping gestoppt wird. Eine mögliche Erklärung dafür ist, dass die durch die Backtranslation erzeugten, Sätze den Originaldaten zu ähnlich sind. Dadurch wirken sie fast wie Duplikate, was es dem Modell erleichtert, die Daten auswendig zu lernen, anstatt sinnvolle Generalisierungen zu entwickeln.

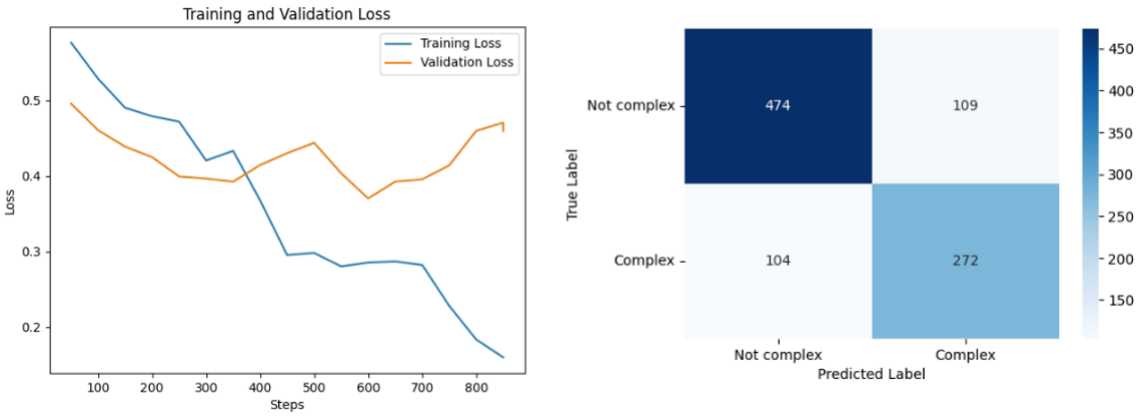


Abbildung 5.4: $\text{GBERT}_{\text{BASE}}$ - Ergebnisse nach Datenaugmentierung mittels Backtranslation

Insbesondere die Erkennung komplexer Wörter verschlechterte sich nach der Anwendung der Backtranslation, wie die Konfusionsmatrix in Abbildung 5.4 zeigt. Dies deutet darauf hin, dass die Vielfalt augmentierten Daten nicht ausreichend war. Tabelle 5.3 zeigt Verschlechterungen aller Metriken im Vergleich zum hyperparameteroptimierten Modell auf.

Modell	F1-Score (macro)	Precision	Recall	Accuracy
$\text{GBERT}_{\text{BASE}}$ (HP-optimiert)	**0.7848**	**0.7823**	**0.7873**	**0.7925**
$\text{GBERT}_{\text{BASE}}$ (HP-optimiert & Datenaugmentierung)	0.7676	0.7670	0.7682	0.7779

Tabelle 5.3: Vergleich der Metriken für $\text{GBERT}_{\text{BASE}}$ (optimiert) und $\text{GBERT}_{\text{BASE}}$ mit Datenaugmentierung

5.4 Ergebnisse und Analyse des Feature-basierten BERT-Ansatzes

Der Verlauf des Trainings- und Validationloss unterscheidet sich zwischen dem Test mit 3 Merkmalen (siehe Abbildung 5.5) und dem mit sieben Merkmalen (siehe Abbildung 5.6) kaum. Beide Modelle lernen sehr schnell auf den Trainingsdaten, was auch hier zu einem Abbruch des Trainings durch Early-Stopping führt, da der Validationloss wieder bei etwa 0.4 stagniert. Die zusätzlichen Informationen, die das Modell durch das Verarbeiten der vier weiteren Merkmale erhalten hat, scheinen jedoch positive Auswirkungen auf die

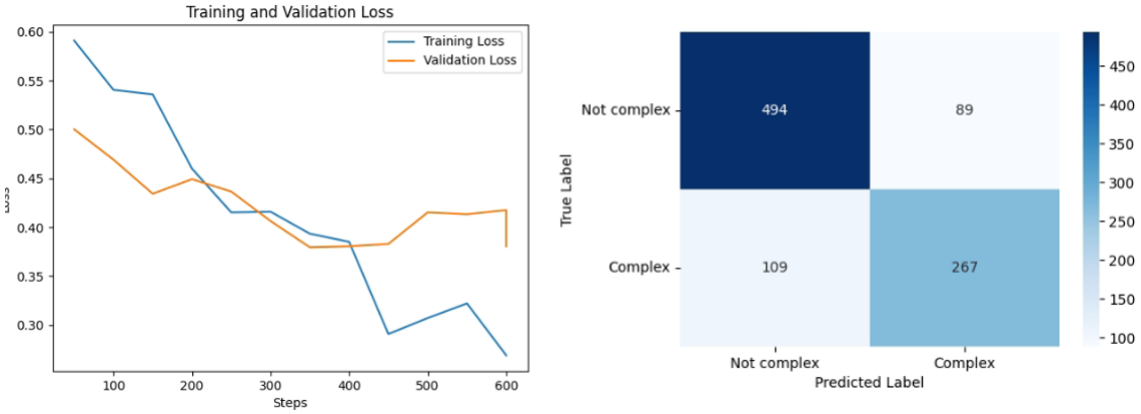


Abbildung 5.5: $\text{GBERT}_{\text{BASE}}$ - Ergebnisse des feature-basierten Ansatzes mit den drei relevantesten Merkmalen

Erkennung komplexer Wörter zu haben: Mit sieben Merkmalen wurden 279 komplexe Wörter korrekt erkannt, während es bei drei Merkmalen 267 komplexe Wörter waren. Allerdings hat sich die Erkennung nicht-komplexer Wörter leicht verschlechtert, von 494 korrekt klassifizierten nicht-komplexen Wörtern bei drei Merkmalen auf 485 Wörter bei sieben Merkmalen. Im Vergleich zum rein hyperparameteroptimierten $\text{GBERT}_{\text{BASE}}$ ist dennoch bei beiden Varianten eine Verschlechterung der Klassifikation komplexer Wörter zu beobachten, was aus dem Recall in Tabelle 5.4 hervorgeht. Dank der leichten Verbesserung bei der Erkennung nicht-komplexer Wörter ist der macro-averaged F1-Score sowie die Genauigkeit bei sieben Merkmalen allerdings sogar leicht gestiegen.

Modell	**F1-Score (macro)**	**Precision**	**Recall**	**Accuracy**
$\text{GBERT}_{\text{BASE}}$ (HP-optimiert)	0.7848	0.7229	**0.7633**	0.7925
$\text{GBERT}_{\text{BASE}}$ (HP-optimiert & 3 Merkmale)	0.7817	0.7500	0.7101	0.7935
$\text{GBERT}_{\text{BASE}}$ (HP-optimiert & 7 Merkmale)	**0.7868**	0.7401	0.7420	**0.7967**

Tabelle 5.4: $\text{GBERT}_{\text{BASE}}$ - Vergleich der feature-basieretn Ansätze mit dem Modell nach der Hyperparameter-Optimierung

Im Vergleich zum RFC aus Kapitel 4.2 zeigt sich, dass die Leistung des BERT-Modells durch das Hinzufügen vieler linguistischer Merkmale verbessert werden kann. Der RFC hingegen kann aus den gewählten Merkmalen kaum Vorteile ziehen, da diese für ihn zu irrelevant oder nur schwach informativ erscheinen. BERT hingegen kann zusätzliche Fea-

Abbildung 5.6: $\text{GBERT}_{\text{BASE}}$ - Ergebnisse des feature-basierten Ansatzes mit sieben Merkmalen

tures besser verarbeiten und von ihnen profitieren, da es durch seine Architektur robust gegenüber hochdimensionalen Daten ist. Mehr Features bieten dem Modell zusätzliche semantische Informationen, was in diesem Fall zu einer leicht besseren Generalisierung führt.

5.5 Einordnung in den CWI 2018 Shared Task

In Tabelle 5.5 ist die Leistung, gemessen an dem macro-averaged F1-Score, der Top-5-Teams des CWI 2018 Shared Task mit den hier erzielten Ergebnissen der BERT-Modelle gegnübergestellt. Es ist zu erkennen, dass keines der Top-5-Teams des CWI 2018 Shared Task die Baseline übertreffen konnte. Alle dort eingesetzten traditionellen Machine-Learning-Modelle wie RFC, SVM oder Ensemble-Methoden lieferten im Vergleich zu den hier vorgestellten BERT-Modellen schwächere Ergebnisse. Im Gegensatz dazu haben alle BERT-Modelle die Baseline übertroffen. Dies könnte vor allem an der bidirektionalen Fähigkeit von BERT liegen, die es dem Modell ermöglicht, den Text sowohl von links nach rechts als auch von rechts nach links gleichzeitig zu lesen und dadurch einen tiefen, kontextuellen Überblick über die Sätze zu erhalten. Selbst das Baseline-Modell $\text{GBERT}_{\text{BASE}}$ übertrifft alle Top-5-Teams des Shared Task. Dies verdeutlicht die Stärke des Pretrainings und die Fähigkeit von BERT, auch ohne weitere Optimierungen ein tiefes Sprachverständnis zu erzeugen.

Die Leistung der BERT-Modelle wurde durch Hyperparameter-Optimierung weiter gesteigert. Optimierte Modelle erzielten F1-Scores, die deutlich über der Baseline und den Ergebnissen aus dem CWI 2018 Shared Task lagen. Eine Kombination mit linguistischen Merkmalen hat die Modellleistung noch weiter verbessert. In diesem Fall wurden nur wenige relevante Merkmale wie Wortlänge, Silbenanzahl und Frequenzdaten aus dem DWDS-Korpus berücksichtigt, was bereits zu einer Verbesserung geführt hat.

Team/System	**F1-Score**	**Rank**
TMU (Kajiwara und Komachi, 2018)	0.7451	1
SB@GU (Alfter und Pilán, 2018)	0.7427	2
hu-berlin (Popović, 2018)	0.6929	3
SB@GU (Alfter und Pilán, 2018)	0.6992	4
CoastalCPH (Bingel und Bjerva, 2018)	0.6619	5
$\text{GBERT}_{\text{BASE}}$ (HP-optimiert & 7 Merkmale)	**0.7868**	-
$\text{GBERT}_{\text{BASE}}$ (HP-optimiert)	0.7848	-
$\text{GBERT}_{\text{BASE}}$ (HP-optimiert & 3 Merkmale)	0.7817	-
$\text{GBERT}_{\text{BASE}}$	0.7744	-
$\text{GBERT}_{\text{BASE}}$ (HP-optimiert & Datenaugmentierung)	0.7676	-
Baseline	0.7546	-

Tabelle 5.5: Vergleich der Ergebnisse der fünf besten Teams bei der binären Klassifikation deutscher Wörter des CWI 2018 Shared Task (Yimam u. a., 2018) mit den hier vorgestellten BERT-Modellen

6 Fazit & Ausblick

RQ1: *Wie gut eignen sich transformerbasierte Ansätze zur Erkennung komplexer Wörter in der deutschen Sprache im Vergleich zu herkömmlichen Ansätzen des maschinellen Lernens?*

Die Ergebnisse dieser Arbeit zeigen, dass die Verwendung transformerbasierter Ansätze, genauer gesagt Encoder-Only-Modelle wie BERT, eine gute Wahl für die Erkennung komplexer Wörter in der deutschen Sprache darstellt. Modelle wie die hier vorgestellten bieten eine starke Alternative zu herkömmlichen Ansätzen des maschinellen Lernens, wie Random Forest oder SVM. Sie konnten die besten Modelle des CWI 2018 Shared Task, die auf traditionellen Verfahren basierten, sowie die bis dahin ungeschlagene Baseline übertreffen. Dies könnte vor allem auf die bidirektionale Fähigkeit von BERT zurückzuführen sein, die es dem Modell ermöglicht, sowohl den links- als auch den rechtsseitigen Kontext eines Wortes zu berücksichtigen, was zu einem besseren Verständnis von Wortbedeutungen führt. Zudem profitieren vortrainierte Modelle wie die hier vorgestellten von riesigen Textmengen, wodurch sie bereits vor dem Finetuning ein tieferes Textverständnis erlangen.

RQ2: *Welche Optimierungstechniken haben positiven Einfluss auf die Leistung der verschiedenen Modelle bei der Erkennung komplexer Wörter in deutschen Texten?*

Die in dieser Arbeit durchgeführten Optimierungen haben gezeigt, dass sich die Leistung von BERT-Modellen durch Hyperparameter-Optimierung signifikant verbessern lässt. Alleine die Anpassung der wichtigen Hyperparameter Lernrate, Batchgröße und Epochenzahl hatte einen positiven Einfluss auf den F1-Score und die Genauigkeit der Modelle. Darüber hinaus konnte die Kombination von BERT mit linguistischen Merkmalen wie Worthäufigkeit, Wortlänge und Silbenanzahl die Klassifikationsleistung der Modelle weiter verbessern. Insbesondere die Nutzung von den in der Arbeit beschriebenen sieben relevanten Merkmalen führte zu einer Verbesserung im Vergleich zu den Modellen mit

weniger Merkmalen.

Trotz guter Leistungen gibt es in Zukunft mehrere Möglichkeiten zur Verbesserung. Eine Verfeinerung der Hyperparameter-Optimierung könnte zu besseren Ergebnissen führen, indem zusätzliche Parameter berücksichtigt und ein größerer Suchraum abgedeckt wird. Besonders bei der Datenaugmentierung konnte der bisherige Ansatz der Backtranslation keine signifikanten Verbesserungen gegenüber der Baseline des $\text{GBERT}_{\text{BASE}}$-Modells erzielen. Alternative Augmentierungstechniken könnten hier möglicherweise bessere Ergebnisse liefern. So könnte die Backtranslation erweitert werden, indem verschiedene Sprachmodelle verwendet oder mehrstufige Rückübersetzungen durchgeführt werden (z. B. Deutsch → Englisch → Französisch → Deutsch), um die Varianz in den Daten zu erhöhen. Auch könnten Zwischenübersetzungen in Sprachen durchgeführt werden, die linguistisch weiter vom Deutschen entfernt sind, wie zum Beispiel Finnisch, Arabisch oder Mandarin. Dies würde helfen, den relativ kleinen Datensatz mit etwa 9000 Samples zu erweitern. Zusätzlich könnte die Erstellung eines neuen Datensatzes mit deutlich mehr Daten eine Möglichkeit bieten, dem Modell mehr Trainingsbeispiele zur Verfügung zu stellen und dessen Generalisierungsfähigkeit zu verbessern. Darüber hinaus könnten zusätzliche Frequenzmerkmale aus anderen Korpora, insbesondere solchen, die speziell für Sprachlerner erstellt wurden (wie zum Beispiel Simple Wikipedia), in Kombination mit den BERT-Embeddings verwendet werden, um die Leistung weiter zu steigern.

Literaturverzeichnis

[pyphen 2024] *Pyphen: A Python module for hyphenation.* https://pypi.org/project/pyphen/. 2024. – Letzter Zugriff:: 2024-09-12

[Ahmed 2023] Ahmed, Nisha: *What is a Confusion Matrix in Machine Learning?* 2023. – URL https://www.datacamp.com/tutorial/what-is-a-confusion-matrix-in-machine-learning. – Letzter Zugriff:: 2024-09-22

[Alfter und Pilán 2018] Alfter, David ; Pilán, Ildikó: SB@ GU at the complex word identification 2018 shared task. In: *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, 2018, S. 315–321. – Letzter Zugriff:: 2024-09-28

[Arjun 2022] Arjun, Sarkar: *All You Need to Know About Attention and Transformers: In-Depth Understanding Part 1.* https://towardsdatascience.com/all-you-need-to-know-about-attention-and-transformers-in-depth-understanding-part-1-552f0b41d021. 2022. – Letzter Zugriff:: 2024-09-06

[Bingel und Bjerva 2018] Bingel, Joachim ; Bjerva, Johannes: Cross-lingual complex word identification with multitask learning. In: *Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications*, 2018, S. 166–174. – Letzter Zugriff:: 2024-09-24

[Buciluǎ u. a. 2006] Buciluǎ, Cristian ; Caruana, Rich ; Niculescu-Mizil, Alexandru: Model compression. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, S. 535–541. – Letzter Zugriff:: 2024-09-29

[Carvalho u. a. 2019] Carvalho, Diogo V. ; Pereira, Eduardo M. ; Cardoso, Jaime S.: Machine learning interpretability: A survey on methods and metrics. In: *Electronics* 8 (2019), Nr. 8, S. 832. – Letzter Zugriff:: 2024-09-22

[Chan u. a. 2020] Chan, Branden ; Schweter, Stefan ; Möller, Timo: German's next language model. In: *arXiv preprint arXiv:2010.10906* (2020). – Letzter Zugriff:: 2024-07-25

[Chen 2018] Chen, Gang: A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation. (2018). – Letzter Zugriff:: 2024-07-25

[De Belder und Moens 2010] De Belder, Jan ; Moens, Marie-Francine: Text simplification for children. In: *Proceedings of the SIGIR workshop on accessible search systems*, 2010, S. 19–26. – Letzter Zugriff:: 2024-09-22

[Devlin u. a. 2018] Devlin, Jacob ; Chang, Ming-Wei ; Lee, Kenton ; Toutanova, Kristina: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *arXiv preprint arXiv:1810.04805* (2018). – URL https://arxiv.org/abs/1810.04805. – Letzter Zugriff:: 2024-09-10

[Digitales Wörterbuch der deutschen Sprache (DWDS) 2024] Digitales Wörterbuch der deutschen Sprache (DWDS): *Worthäufigkeit*. 2024. – URL https://www.dwds.de/d/worthaeufigkeit. – Letzter Zugriff:: 2024-09-17

[Feng u. a. 2021] Feng, Steven Y. ; Gangal, Varun ; Wei, Jason ; Chandar, Sarath ; Vosoughi, Soroush ; Mitamura, Teruko ; Hovy, Eduard: A Survey of Data Augmentation Approaches for NLP. In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online : Association for Computational Linguistics, August 2021, S. 968–988. – URL https://aclanthology.org/2021.findings-acl.84. – Letzter Zugriff:: 2024-09-29

[Geyken 2007] Geyken, Alexander: The DWDS Corpus: A reference corpus for the German language of the 20th century. In: Fellbaum, Christiane (Hrsg.): *Collocations and Idioms: Linguistic, lexicographic, and computational aspects*, Continuum Press, 2007, S. 23–40. – URL https://www.dwds.de/dwds_static/publications/text/geyken_2007_kerncorpus.pdf. – Letzter Zugriff:: 2024-09-17

[Gooding u. a. 2021] Gooding, Sian ; Kochmar, Ekaterina ; Yimam, Seid M. ; Biemann, Chris: Word Complexity is in the Eye of the Beholder. In: Toutanova, Kristina (Hrsg.) ; Rumshisky, Anna (Hrsg.) ; Zettlemoyer, Luke (Hrsg.) ; Hakkani-Tur, Dilek (Hrsg.) ; Beltagy, Iz (Hrsg.) ; Bethard, Steven (Hrsg.) ; Cotterell, Ryan (Hrsg.) ; Chakraborty, Tanmoy (Hrsg.) ; Zhou, Yichao (Hrsg.): *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online : Association for Computational

Linguistics, 2021, S. 4439–4449. – URL https://aclanthology.org/2021.naacl-main.351. – Letzter Zugriff:: 2024-09-22

[Hinton 2015] Hinton, Geoffrey: Distilling the Knowledge in a Neural Network. In: *arXiv preprint arXiv:1503.02531* (2015). – Letzter Zugriff:: 2024-09-29

[Hochreiter und Schmidhuber 1997] Hochreiter, Sepp ; Schmidhuber, Jürgen: Long Short-Term Memory. In: *Neural Computation* 9 (1997), Nr. 8, S. 1735–1780. – Letzter Zugriff:: 2024-07-25

[Horn u. a. 2014] Horn, Colby ; Manduca, Cathryn ; Kauchak, David: Learning a lexical simplifier using Wikipedia. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2014, S. 458–463. – Letzter Zugriff:: 2024-09-22

[HuggingFace 2024] HuggingFace: *Natural Language Processing Course - Chapter 1: Transformer Models*. https://huggingface.co/learn/nlp-course/chapter1. 2024. – Letzter Zugriff:: 2024-09-10

[Jalammar 2018] Jalammar, Jay A.: *The Illustrated Transformer*. https://jalammar.github.io/illustrated-transformer/. 2018. – Letzter Zugriff:: 2024-08-06

[Junczys-Dowmunt u. a. 2018] Junczys-Dowmunt, Marcin ; Grundkiewicz, Roman ; Dwojak, Tomasz ; Hoang, Hieu ; Heafield, Kenneth ; Neckermann, Tom ; Seide, Frank ; Germann, Ulrich ; Fikri Aji, Alham ; Bogoychev, Nikolay ; Martins, André F. T. ; Birch, Alexandra: Marian: Fast Neural Machine Translation in C++. In: *Proceedings of ACL 2018, System Demonstrations*. Melbourne, Australia : Association for Computational Linguistics, July 2018, S. 116–121. – URL http://www.aclweb.org/anthology/P18-4020. – Letzter Zugriff:: 2024-10-02

[Kajiwara und Komachi 2018] Kajiwara, Tomoyuki ; Komachi, Mamoru: Complex word identification based on frequency in a learner corpus. In: *Proceedings of the thirteenth workshop on innovative use of NLP for Building Educational Applications*, 2018, S. 195–199. – Letzter Zugriff:: 2024-09-24

[Kauchak 2013] Kauchak, David: Improving text simplification language modeling using unsimplified text data. In: *Proceedings of the 51st annual meeting of the association for computational linguistics (volume 1: Long papers)*, 2013, S. 1537–1546. – Letzter Zugriff:: 2024-09-22

[Knollman-Porter u. a. 2015] Knollman-Porter, Kelly ; Wallace, Sarah E. ; Hux, Karen ; Brown, Jessica ; Long, Candace: Reading experiences and use of supports by people with chronic aphasia. In: *Aphasiology* 29 (2015), Nr. 12, S. 1448–1472. – Letzter Zugriff:: 2024-09-22

[Naber 2005] Naber, Daniel: OpenThesaurus: ein offenes deutsches Wortnetz. In: *Sprachtechnologie, mobile Kommunikation und linguistische Ressourcen: Beiträge zur GLDV-Tagung* (2005), S. 422–433. – Letzter Zugriff:: 2024-09-17

[Olah 2015] Olah, Christopher: *Understanding LSTM Networks*. 2015. – URL https://colah.github.io/posts/2015-08-Understanding-LSTMs/. – Letzter Zugriff:: 2024-07-05

[Paetzold und Specia 2016] Paetzold, Gustavo H. ; Specia, Lucia: SemEval 2016 Task 11: Complex Word Identification. In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)* (2016), S. 560–569. – Letzter Zugriff:: 2024-07-25

[Pan u. a. 2021] Pan, Chunguang ; Song, Bingyan ; Wang, Shengguang ; Luo, Zhipeng: DeepBlueAI at SemEval-2021 Task 1: Lexical complexity prediction with a deep ensemble approach. In: *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, 2021, S. 578–584. – Letzter Zugriff:: 2024-09-28

[Popović 2018] Popović, Maja: Complex word identification using character n-grams. In: *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, 2018, S. 341–348. – Letzter Zugriff:: 2024-09-29

[Sanh 2019] Sanh, V: DistilBERT, A Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. In: *arXiv preprint arXiv:1910.01108* (2019). – Letzter Zugriff:: 2024-09-23

[Sedgwick 2012] Sedgwick, Philip: Pearson's correlation coefficient. In: *Bmj* 345 (2012). – Letzter Zugriff:: 2024-09-28

[Sennrich u. a. 2016] Sennrich, Rico ; Haddow, Barry ; Birch, Alexandra: Improving Neural Machine Translation Models with Monolingual Data. In: Erk, Katrin (Hrsg.) ; Smith, Noah A. (Hrsg.): *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany : Association for Computational Linguistics, August 2016, S. 86–96. – URL https://aclanthology.org/P16-1009. – Letzter Zugriff:: 2024-09-29

[Shardlow 2013a] Shardlow, Matthew: A Comparison of Techniques to Automatically Identify Complex Words. In: *51st annual meeting of the association for computational linguistics proceedings of the student research workshop*, 2013, S. 103–109. – Letzter Zugriff:: 2024-09-22

[Shardlow 2013b] Shardlow, Matthew: The cw corpus: A new resource for evaluating the identification of complex words. In: *Proceedings of the Second Workshop on Predicting and Improving Text Readability for Target Reader Populations*, 2013, S. 69–77. – Letzter Zugriff:: 2024-09-22

[Shardlow 2014] Shardlow, Matthew: Out in the Open: Finding and Categorising Errors in the Lexical Simplification Pipeline. In: *LREC*, 2014, S. 1583–1590. – Letzter Zugriff:: 2024-09-22

[Shardlow u. a. 2020] Shardlow, Matthew ; Cooper, Michael ; Zampieri, Marcos: Complex: A new corpus for lexical complexity prediction from likert scale data. In: *arXiv preprint arXiv:2003.07008* (2020). – Letzter Zugriff:: 2024-09-28

[Shardlow u. a. 2021] Shardlow, Matthew ; Evans, Richard ; Paetzold, Gustavo H. ; Zampieri, Marcos: Semeval-2021 task 1: Lexical complexity prediction. In: *arXiv preprint arXiv:2106.00473* (2021). – Letzter Zugriff:: 2024-09-28

[Suárez u. a. 2019] Suárez, Pedro Javier O. ; Sagot, Benoît ; Romary, Laurent: Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures. In: *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)* Leibniz-Institut für Deutsche Sprache (Veranst.), 2019. – Letzter Zugriff:: 2024-09-22

[Vaswani u. a. 2017] Vaswani, Ashish ; Shazeer, Noam ; Parmar, Niki ; Uszkoreit, Jakob ; Jones, Llion ; Gomez, Aidan N. ; Kaiser, Łukasz ; Polosukhin, Illia: Attention is all you need. In: *Advances in Neural Information Processing Systems* (2017). – Letzter Zugriff:: 2024-08-16

[Wei und Zou 2019] Wei, Jason ; Zou, Kai: Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In: *arXiv preprint arXiv:1901.11196* (2019). – Letzter Zugriff:: 2024-09-29

[Wiegand u. a. 2018] Wiegand, Michael ; Siegel, Melanie ; Ruppenhofer, Josef: Overview of the germeval 2018 shared task on the identification of offensive language. (2018). – Letzter Zugriff:: 2024-08-06

[Xu u. a. 2020] Xu, Peng ; Kumar, Dhruv ; Yang, Wei ; Zi, Wenjie ; Tang, Keyi ; Huang, Chenyang ; Cheung, Jackie Chi K. ; Prince, Simon J. ; Cao, Yanshuai: Optimizing deeper transformers on small datasets. In: *arXiv preprint arXiv:2012.15355* (2020). – Letzter Zugriff:: 2024-09-29

[Yamada u. a. 2020] Yamada, Ikuya ; Asai, Akari ; Sakuma, Jin ; Shindo, Hiroyuki ; Takeda, Hideaki ; Takefuji, Yoshiyasu ; Matsumoto, Yuji: Wikipedia2Vec: An Efficient Toolkit for Learning and Visualizing the Embeddings of Words and Entities from Wikipedia. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Association for Computational Linguistics, 2020, S. 23–30. – Letzter Zugriff:: 2024-09-17

[Yang und Shami 2020] Yang, Li ; Shami, Abdallah: On hyperparameter optimization of machine learning algorithms: Theory and practice. In: *Neurocomputing* 415 (2020), S. 295–316. – URL https://www.sciencedirect.com/science/article/pii/S0925231220311693. – Letzter Zugriff:: 2024-10-02. – ISSN 0925-2312

[Yaseen u. a. 2021] Yaseen, Tuqa B. ; Ismail, Qusai ; Al-Omari, Sarah ; Al-Sobh, Eslam ; Abdullah, Malak: Just-blue at semeval-2021 task 1: Predicting lexical complexity using bert and roberta pre-trained language models. In: *Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021)*, 2021, S. 661–666. – Letzter Zugriff:: 2024-09-28

[Yimam u. a. 2018] Yimam, Seid M. ; Biemann, Chris ; Malmasi, Shervin ; Paetzold, Gustavo ; Specia, Lucia ; Štajner, Sanja ; Tack, Anaïs ; Zampieri, Marcos: A Report on the Complex Word Identification Shared Task 2018. In: Tetreault, Joel (Hrsg.) ; Burstein, Jill (Hrsg.) ; Kochmar, Ekaterina (Hrsg.) ; Leacock, Claudia (Hrsg.) ; Yannakoudakis, Helen (Hrsg.): *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*. New Orleans, Louisiana : Association for Computational Linguistics, Juni 2018, S. 66–78. – URL https://aclanthology.org/W18-0507. – Letzter Zugriff:: 2024-10-02

[Yimam u. a. 2017] Yimam, Seid M. ; Biemann, Chris ; Roegiest, Arjen ; Thomas, Alexandra ; Saggion, Horacio ; Specia, Lucia: Multilingual and Cross-Lingual Complex Word Identification. In: *Transactions of the Association for Computational Linguistics* 5 (2017), S. 341–356. – Letzter Zugriff:: 2024-07-25

[Ying 2019] Ying, Xue: An overview of overfitting and its solutions. In: *Journal of physics: Conference series* Bd. 1168, 2019, S. 022022. – Letzter Zugriff:: 2024-09-29

[Zhang u. a. 2019] Zhang, Xingjian ; Qu, Lixia ; Tresp, Volker: Recurrent Neural Networks (RNNs): A Gentle Introduction and Overview. In: *arXiv preprint arXiv:1912.05911* (2019). – Letzter Zugriff:: 2024-07-25

[Zhu 2015] Zhu, Yukun: Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books. In: *arXiv preprint arXiv:1506.06724* (2015). – Letzter Zugriff:: 2024-07-24

[Zhuang u. a. 2021] Zhuang, Fuzhen ; Qi, Zhiyuan ; Duan, Keyu ; Xi, Dongbo ; Zhu, Yongchun ; Zhu, Hengshu ; Xiong, Hui ; He, Qing: A Comprehensive Survey on Transfer Learning. In: *Proceedings of the IEEE* (2021). – Letzter Zugriff:: 2024-07-25

A Anhang

A.1 Abbildungen

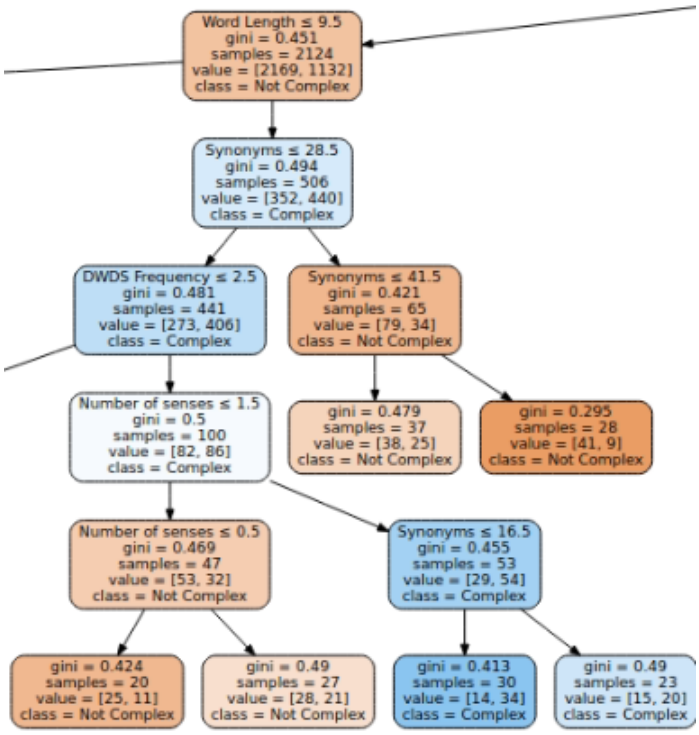
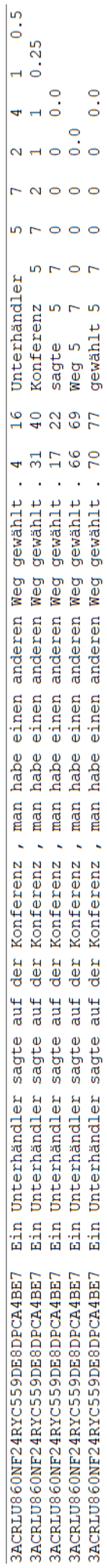


Abbildung A.1: Ausschnitt aus einem Entscheidungsbaum des RFC



3ACRLU860NF24RYC559DE8DPCA4BE7	Ein Unterhändler sagte auf der Konferenz , man habe einen anderen Weg gewählt .	4	16	Unterhändler	5	7	2	4	1	0.5
3ACRLU860NF24RYC559DE8DPCA4BE7	Ein Unterhändler sagte auf der Konferenz , man habe einen anderen Weg gewählt .	31	40	Konferenz	5	7	2	1	1	0.25
3ACRLU860NF24RYC559DE8DPCA4BE7	Ein Unterhändler sagte auf der Konferenz , man habe einen anderen Weg gewählt .	17	22	sagte	5	7	0	0	0	0.0
3ACRLU860NF24RYC559DE8DPCA4BE7	Ein Unterhändler sagte auf der Konferenz , man habe einen anderen Weg gewählt .	66	69	Weg	5	7	0	0	0	0.0
3ACRLU860NF24RYC559DE8DPCA4BE7	Ein Unterhändler sagte auf der Konferenz , man habe einen anderen Weg gewählt .	70	77	gewählt	5	7	0	0	0	0.0

Abbildung A.2: Beispiel Trainingsdatensatz

Erklärung zur selbständigen Bearbeitung

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

Ort	Datum	Unterschrift im Original