

BACHELOR THESIS
Kristoffer Schaaf

Entwicklung einer Software zur Erkennung von Fake News auf Nachrichtenportalen

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

Kristoffer Schaaf

Entwicklung einer Software zur Erkennung von Fake News auf Nachrichtenportalen

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

Betreuer Prüfer: Prof. Dr. Stefan Sarstedt
Zweitgutachter: Prof. Dr. Marina Tropmann-Frick

Eingereicht am: 01.04.2025

Kristoffer Schaaf

Thema der Arbeit

Entwicklung einer Software zur Erkennung von Fake News auf Nachrichtenportalen

Stichworte

Machinelles Lernen, Fake News, Nachrichtenportale,

Kurzzusammenfassung

Arthur Dents Reise in eine neue Zukunft . . .

Kristoffer Schaaf

Title of Thesis

Development of a software for the detection of fake news on news portals

Keywords

Machine Learning, Fake News, Text Mining, Classification, NLP

Abstract

Arthur Dents travel to a new future . . .

Inhaltsverzeichnis

Abbildungsverzeichnis	vi
Tabellenverzeichnis	viii
1 Einleitung	1
1.1 Hintergrund: Die zunehmende Verbreitung von Fake News und deren gesellschaftliche Auswirkungen	1
1.2 Zielsetzung: Entwicklung einer Software zur automatisierten Fake-News-Erkennung	5
1.3 Wahl der Nachrichtenportale	5
1.4 Aufbau der Arbeit	6
2 Natural Language Processing	7
2.1 Machine Learning	7
2.1.1 Textbereinigung und Vorverarbeitung	7
2.1.2 Merkmalsextraktion	9
2.1.3 Machine Learning Modelle	13
2.2 Deep Learning	23
2.2.1 Word Embeddings	23
2.2.2 Deep Learning Modelle	26
2.3 Transformer	28
2.3.1 Grundlagen der Transformer Architektur	28
2.3.2 Hidden Layer	31
2.3.3 Transformer Modelle	32
2.4 Metriken	38
2.5 Hybride Modelle zur Fake News Erkennung	40
3 Relevante Datensätze und Auswahlkriterien	51
3.1 Vorstellung verfügbarer deutscher Fake-News-Datensätze	51

3.2	Nutzung englischer Datensätze	51
3.3	Auswahl und Begründung des finalen Datensatzes	52
4	Konzeption der Softwarelösung	55
4.1	Machine Learning Modell	55
4.1.1	Auswahl und Begründung der genutzten Modelle	55
4.1.2	Datenvorverarbeitung	56
4.1.3	Fine-tuning der Transfomer Modelle	56
4.1.4	Erzeugung der Embeddings	57
4.1.5	Nutzung der Embeddings im LightGBM Modell	58
4.1.6	Gesamtarchitektur	58
4.2	Webagent	58
5	Umsetzung des Prototyps	62
5.1	Implementierung des Machine Learning Modells	62
5.1.1	Fine-tuning der Transfomer Modelle	62
5.1.2	Erzeugung der Embeddings	63
5.1.3	Nutzung der Embeddings im LightGBM Modell	64
5.1.4	Gesamtarchitektur	65
5.2	Implementierung der Chrome Erweiterung	65
6	Evaluation und Ergebnisse	67
7	Fazit	68
8	Ausblick	69
Literaturverzeichnis		70
A	Anhang	82
A.1	Verwendete Hilfsmittel	82
A.2	Abbildungen	82
A.3	Tabellen	82
Selbstständigkeitserklärung		90

Abbildungsverzeichnis

2.1	Vergleich der Sparse Matrizen	9
2.2	Vergleich verschiedener Modelle mit BoW, TF-IDF und Hashing [6]	12
2.3	Darstellung von Hyperplanes [45]	16
2.4	Auswertung verschiedener NLP Algorithmen [67]	21
2.5	XGBoost	22
2.6	Bsp. für Word Embeddings in einem dreidimensionalen Vektorraums [9] . .	24
2.7	Vergleich CBOW (links) und Skip-gram (rechts) [57]	25
2.8	Co-Occurrence-Wahrscheinlichkeiten für die Zielwörter „ice“ und „steam“ mit ausgewählten Kontextwörtern aus einem Korpus mit 6 Milliarden Tokens [65]	26
2.9	Vgl. RNN (links) und LSTM (rechts) [2]	27
2.10	Vgl. LSTM und BiLSTM [78]	28
2.11	Eine Übersicht der Transformer Architektur [49] - vereinfacht von [85] . . .	29
2.12	Bsp. zum MLM [91]	33
2.13	Bidirektionality des BERT Modells [88]	33
2.14	Zusammensetzung eines Input Tokens im BERT Modell [25]	35
2.15	Auswertung eines bilingualen Satzpaars von TLM [18]	37
2.16	Konfusionsmatrix	39
2.17	Ergebnisse der Klassifizierungen der verschiedenen CNN-LSTM Modelle [83]	40
2.18	Ergebnisse verschiedener Modelle mit GloVe Embedding [12]	41
2.19	Ergebnisse der verschiedenen Modelle bei Validierung [47]	43
2.20	Ergebnisse der verschiedenen Modelle [87]	44
2.21	Architekture des vorgeschlagenen hybriden Modells [69]	44
2.22	Ergebnisse der verschiedenen Modelle mit dem PolitiFact Datensatz [69] .	45
2.23	Architektur des hybriden Modells [89]	46
2.24	Ergebnisse der verschiedenen Modelle [89]	47
2.25	Architektur des hybriden Modells [31]	48
2.26	Ergebnisse der verschiedenen Modelle auf dem FNC-Datensatz [31]	48

Abbildungsverzeichnis

2.27 Vergleich der Ergebnisse der verschiedenen Modelle [84]	50
4.1 Beispielhafter Ablauf einer Klassifizierung eines Artikels	59
5.1 K-Fold-Cross-Validation [61]	64
A.1 Architektur des BERT Modells [64]	83
A.2 Multi-Head Attention [64]	84
A.3 Sequenzdiagramm Webagent	85

Tabellenverzeichnis

2.1	Vergleich der Vor- und Nachteile von BoW und TF-IDF	12
2.2	Übersicht von Aktivierungsfunktion in der logistischen Regression [46, 39, 1]	18
3.1	Vergleich deutscher Fake-News-Datensätze	52
6.1	Direktes Fine-Tuning – Testergebnisse der Modelle	67
6.2	LightGBM-Ergebnisse auf Embeddings der Modelle	67
6.3	Accuracy und F1-Score: Transformer Fine-Tuning vs. LightGBM auf Embeddings	67
A.1	Verwendete Hilfsmittel und Werkzeuge	82
A.2	Vergleich möglicher Technologien für den Webagenten	86
A.3	Vergleich der verschiedenen BERT- und RoBERTa-Modelle	87
A.4	Überblick über die gewählten Hyperparameter der Transformer-Modelle .	88
A.5	Überblick über die gewählten Hyperparameter des LightGBM-Modells .	89

1 Einleitung

1.1 Hintergrund: Die zunehmende Verbreitung von Fake News und deren gesellschaftliche Auswirkungen

1.1.1 Wann entstanden Fake News

Fake News sind ein allgegenwärtiges Problem, doch hatten Sie Ihren ersten Auftritt bereits 44BC im römischen Reich [8]. Auch während des amerikanischen Bürgerkriegs 1779 wurden Sie als politischer Schachzug von Benjamin Franklin genutzt. Dieser schickte einen Brief an Captain Samuel Gerrish und schrieb in diesem über Grausamkeiten der Briten und deren Verbündeten. Diese Informationen wurde so veranschaulicht, dass sie die öffentliche Meinung bewusst beeinflussen sollten [77].

Der eigentliche Begriff "Fake News" wurde erst viele Jahre später durch Donald Trump im amerikanischen Wahlkampf 2016 bekannt [5] und diente hierbei als politischer Kampfbegriff [13].

Unter anderem ist Fake News auch ein Teil von Propaganda [13], welche schon lange als Mittel zur Meinungsmanipulation eines Volkes genutzt wird.

Heute ist Fake News die größte Drohung zu unserer angeblich freien Presse [77].

1.1.2 Wie definieren sich Fake News und wie sind sie aufgebaut

Fake News sind bewusst erstellte Online-Falschmeldungen, die teilweise oder vollständig unwahre Inhalte verbreiten, um Leser*innen gezielt zu täuschen oder zu manipulieren. Sie imitieren klassische Nachrichtenformate, nutzen auffällige Titel, emotionale Bilder und strategisch gestaltete Inhalte, um Glaubwürdigkeit zu erzeugen und Aufmerksamkeit zu

gewinnen. Ziel ist es, durch das Verbreiten dieser Inhalte Klicks, Reichweite und damit finanzielle oder ideologische Vorteile zu erzielen [8].

Fake News fallen in die Kategorien Satire, Clickbait, Gerüchte, Stance News, Propaganda und Large Scale Hoaxes [77].

- **Satire:** ist eine humorvolle oder übertriebene Darstellung gesellschaftlicher oder politischer Themen, die Kritik üben soll.
- **Clickbait:** bezeichnet reißerische Überschriften oder Vorschaubilder, die Neugier wecken und zum Anklicken eines Inhalts verleiten sollen, oft ohne den Erwartungen gerecht zu werden.
- **Gerüchte:** sind unbestätigte Informationen, die sich schnell verbreiten und oft falsch oder irreführend sind.
- **Stance News:** sind Nachrichten, die eine klare Meinung oder politische Haltung einnehmen, statt neutral zu berichten.
- **Propaganda:** ist die gezielte Verbreitung von Informationen oder Meinungen, um das Denken und Handeln von Menschen zu beeinflussen, meist im Interesse einer bestimmten Gruppe oder Ideologie.
- **Large Scale Hoaxes:** sind absichtlich erfundene Falschmeldungen oder Täuschungen, die weit verbreitet werden und viele Menschen täuschen sollen.

Die eigentliche Nachricht ist aufgebaut in folgende Teile:

- **Quelle:** gibt den Ersteller der Nachricht an.
- **Titel:** erzielt die Aufmerksamkeit der Lesenden.
- **Text:** enthält die eigentliche Information der Nachricht.
- **Medien:** in Form von Bildern oder Videos.

Fake News können die Form von Text, Fotos, Filmen oder Audio annehmen und sind dementsprechend auf jeder Platform auffindbar, die die Verbreitung nicht unterbindet. Die 2024 populärste Platform zum Teilen der Fake News ist WhatsApp [5].

1.1.3 Aus welcher Motivation entstehen Fake News

Das Hauptinteresse der Ersteller der Fake News ist das Verdienen von Geld. Auf die Artikel wird Werbung geschaltet und anhand einer entsprechenden Reichweite ergibt sich der verdiente Betrag. Je mehr Reichweite, desto mehr Verdienst für die Ersteller [8].

1.1.4 Warum verbreiten sich Fake News

In sozialen Medien neigen Nutzer aufgrund von FOMO (Fear of Missing Out) dazu, Fake News zu teilen, um Anerkennung zu gewinnen und soziale Zugehörigkeit zu erfahren. Besonders häufig werden kontroverse, überraschende oder bizarre Inhalte verbreitet – insbesondere dann, wenn sie starke Emotionen wie Freude, Wut oder Aufregung hervorrufen. Das Teilen solcher Inhalte stärkt das eigene Ansehen, da es signalisiert, über neue und relevante Informationen zu verfügen. Fake News bestehen meist aus eindrucksvoll präsentierten Falschinformationen [8].

Ein Grund für die schnelle Verbreitung von Fake News liegt in ihrer Aufmachung: Häufig wird die zentrale Aussage bereits in der Überschrift formuliert, oft mit Bezug auf konkrete Personen oder Ereignisse. Dadurch überspringen viele Leser den Artikel selbst, was die Wirkung von Schlagzeilen verstärkt. Die Inhalte sind meist kurz, wiederholend und wenig informativ. Anders als bei seriösen Nachrichten, bei denen Argumente überzeugen sollen, wirken Fake News über einfache Denkabkürzungen (Heuristiken) und die Bestätigung bestehender Überzeugungen. Nutzer müssen sich also nicht mit komplexen Inhalten auseinandersetzen, sondern lassen sich durch intuitive Übereinstimmungen überzeugen. Besonders bei geringer kognitiver Anstrengung – etwa durch Müdigkeit oder Unaufmerksamkeit – steigt die Wahrscheinlichkeit, dass Fake News geglaubt und weiterverbreitet werden [42].

1.1.5 Wer konsumiert Fake News

Laut [42] sind folgende Gruppen die größten Konsumenten:

- **Geringe Bildung oder digitale Kompetenz:** Personen mit niedriger formaler Bildung oder unzureichenden digitalen Fähigkeiten sind anfälliger für Falschinformationen.

- **Aussagen oder Nähe zur Informationsquelle:** Informationen von Personen, denen man persönlich nahe steht oder vertraut, werden eher geglaubt – unabhängig vom Wahrheitsgehalt.
- **Parteizugehörigkeit oder politische Überzeugung:** Menschen neigen dazu, Fake News zu glauben und zu verbreiten, wenn diese mit ihrer ideologischen Einstellung übereinstimmen.
- **Misstrauen gegenüber den Medien:** Wer etablierten Medien nicht vertraut, ist eher bereit, alternative (oftmals falsche) Quellen zu konsumieren und zu verbreiten.
- **Geringere kognitive Fähigkeiten:** Personen mit niedrigerer kognitiver Verarbeitungskapazität sind anfälliger für einfache, irreführende Inhalte und hinterfragen diese seltener kritisch.

Außerdem scheinen konservative, rechtsgerichtete Menschen, ältere Personen und weniger gebildete Menschen eher dazu zu neigen, Fake News zu glauben und zu verbreiten [8].

1.1.6 Welche potenziellen Indikatoren zum Erkennen bei Fake News gibt es

Das Erkennen von Fake News ist gerade deshalb problematisch, da diese erst erkannt werden können, nachdem sie erstellt und im Internet verbreitet wurden. [77]

Gerade im Bereich der sozialen Medien gibt es aber relativ zuverlässige Indikatoren, die Fake News nach der Erstellung als solche zu enttarnen [38]:

- **Fortlaufende Großschreibung:** Beispiel: GROßSCHREIBUNG
- **Übermäßige Nutzung von Satzzeichen:** Beispiel: !!!
- **Falsche Zeichensetzung am Satzende:** Beispiel: !!1
- **Übermäßige Nutzung von Emoticons, besonders auffälliger Emoticons**
- **Nutzung des Standard-Profilbildes**
- **Fehlende Account-Verifizierung, besonders bei prominenten Personen**

Fake News in offiziellen Nachrichtenportalen zu erkennen, ist dagegen deutlich schwieriger. Die aufgezählten stilistischen Mittel wie zum Beispiel die fortlaufende Großschreibung sind eher untypisch. Stattdessen muss über die inhaltliche Bedeutung erkannt werden ob die Artikel wahr oder falsch sind.

1.2 Zielsetzung: Entwicklung einer Software zur automatisierten Fake-News-Erkennung

Motiviert durch die Arbeiten der University of Applied Sciences Upper Austria [79] und der TU Darmstadt [38] wird in dieser Arbeit die Entwicklung eines weiteren Tools dokumentieren. Dieses Tool soll wie auch das Browser Plugin TrustyTweet eine Unterstützung zum Erkennen von Fake News anbieten. Ob dieses Tool auch als Browser Plugin oder als eine andere Form der Software implementiert wird, steht zum jetzigen Zeitpunkt noch nicht fest. Auch ob eine Black Box oder White Box Architektur genutzt wird, - das heißt, kann der User zum Beispiel sehen, warum der Artikel als Fake News deklariert wird - wird im Laufe der Arbeit entschieden. Ziel ist es, dass das Tool nicht wie TrustyTweet auf Twitter eingesetzt wird, sondern auf verschiedenen Nachrichtenportalen. Um eine politisch möglichst breite Abdeckung zu decken, wird das Tool für drei verschiedenen Nachrichtenportalen implementiert (siehe Kapitel 1.3).

1.3 Wahl der Nachrichtenportale

Im Paper der University of Applied Sciences Upper Austria [79] wird die Qualität verschiedener deutscher Nachrichtenportale mit „Machine Learning“-Modellen getestet. Das Ergebnis zeigt, dass Spiegel, Die Zeit und Süddeutsche die ’besten’ Portale sind. Express, BZ-Berlin und Bild sind die ’schlechtesten’, da sie am meisten Fake News verbreiten. Die Quellen [40, 53, 62] zeigen, dass die Kombination von BILD, taz und Der Spiegel eine politisch breites Spektrum abbilden.

- **BILD:** Boulevarddesk, populistisch, konservativ

Die BILD-Zeitung gilt als stark meinungsgtriebenes Boulevardmedium mit populistischen Zügen. Ihre Berichterstattung ist geprägt von einer emotionalisierenden Sprache, Fokus auf Einzelereignisse und dem Ziel hoher Reichweiten.

- **taz:** Kritisch, linksalternativ, bewegungsnah

Die taz (tageszeitung) wird dem linksalternativen Spektrum zugeordnet. Sie verfolgt eine aktivistische Grundhaltung mit einem Fokus auf sozialen Bewegungen, Umweltfragen und Minderheitenrechten. Die taz gilt als Gegenmodell zu großen Leitmedien und strebt oft bewusst Gegenöffentlichkeit an.

- **Der Spiegel:** Linksliberal, investigativ, kritisch gegenüber Macht

Der Spiegel wird dem linksliberalen Spektrum zugeordnet. Er kombiniert klassische Leitmedienformate mit einem ausgeprägten Anspruch auf investigativen Journalismus, Kritik an staatlicher Macht und liberal-demokratischen Werten.

Das Tool, welches in dieser Arbeit entwickelt wird, wird also für diese drei Nachrichtenportale implementiert.

1.4 Aufbau der Arbeit

2 Natural Language Processing

2.1 Machine Learning

2.1.1 Textbereinigung und Vorverarbeitung

- **Titel und Inhalt der Artikel zusammenfügen [12]**: Damit keine wichtigen Informationen verloren gehen, werden Titel und Inhalt des Artikels zusammengefasst. Gerade der Titel kann durch z.B. Clickbait (siehe 1.1.2) schnell Hinweise auf eventuelle Fake News geben.
- **Akzente und Sonderzeichen entfernen [12] [71] [6]**: Akzente führen dazu, dass Wörter wie „café“ und „cafe“ unterschiedlich behandelt werden, obwohl sie semantisch gleich sind. Das Entfernen dieser erhöht die Generalisierung. Sonderzeichen stören einfache Tokenizer (z. B. bei Bag-of-Words), führen zu vielen seltenen Tokens und zu überdimensionierten Vektoren (siehe 2.1.2).
- **Alle Buchstaben zu Kleinbuchstaben konvertieren [71] [80] [6]**: Ähnlich wie zum vorherigen Punkt erhöht die durchgehende Kleinschreibung aller Buchstaben die Generalisierung und verhindert somit unnötige Duplikate im Vokabular.
- **Leere Spalten entfernen [80]**: Leere Spalten enthalten keine Information. Sie können bei der Vektorisierung oder Modellerstellung Fehler verursachen und werden als einfache Datenbereinigungsmaßnahme entfernt.
- **Kontraktionen auflösen (ans -> an das) [12]**: Im deutschen sind Kontradiktionen zwar nicht so häufig wie im englischen, sie kommen aber trotzdem vor und sollten aufgelöst werden. Dies vermeidet fragmentierte Token und verbessert die Semantik und Trennbarkeit im Modell.

- **Stoppwörter entfernen [12] [71] [6]:** Wörter wie „der“, „ist“, „und“ tragen wenig zur inhaltlichen Differenzierung bei. Das Entfernen dieser verbessert die semantische Gewichtung relevanter Begriffe [72].
- **Rechtschreibfehler korrigieren [71]:** Tippfehler führen zu seltenen Tokens und stören die Generalisierung. In offiziellen Artikeln sind zwar selten Rechtschreibfehler zu finden, aber falls vorhanden, hilft die Korrektur zur Verbesserung der Modellqualität.
- **Lemmatisieren [12] [71] [6]:** Bei der Lemmatisierung werden verschiedene Wortformen auf die Grundform zurückgeführt („läuft“, „lief“, „laufen“ wird zu „laufen“). So erkennt das Modell gleiche Bedeutungen trotz grammatischer Variation.
- **Tokenisierung [71]:** In der Tokenisierung werden die Texte in einzelne Wörter oder Einheiten (Tokens) zerlegt, die für Modelle verarbeitbar sind. Dies ist eine Grundvoraussetzung für alle weiteren NLP-Schritte wie TF-IDF oder Word Embeddings.

Nutzung einer dualen Feature-Pipeline

Ein Problem welches das Entfernen der Akzente und Sonderzeichen und das Konvertieren aller Buchstaben zu Kleinbuchstaben mit sich bringt ist, dass viele wichtige Hinweise zum Erkennen von Fake News verloren gehen. Wie in Kapitel 1.1.6 beschrieben, sind fortlaufende Großschreibung, übermäßige Nutzung von Satzzeichen und falsche Zeichensetzung am Satzende potenzielle Indikatoren für Fake News.

Eine duale Feature-Pipeline kann dieses Problem lösen. Implementiert wird eine „cleaned“ Version (z.B. für inhaltliche Bedeutung) mit standardisierten, inhaltlichen Features und eine „rohe“ Version (z.B. für Stilmerkmale) mit stilistischen, rohen Textfeatures.

So werden semantische und stilistische Hinweise genau so genutzt wie ein Mensch es beim Lesen macht.

Die Notwendigkeit der Stilmerkmale ist aber diskutierbar. Die Datensätze werden ausschließlich aus Artikeln von offiziellen Nachrichtenportalen zusammengesetzt. Diese schreiben meist sauber, ohne Caps-Lock oder auffällige Sonderzeichen. Stilistische Merkmale wie viele Ausrufezeichen, Emojis oder absichtliche Rechtschreibfehler kommen dort nicht vor – also sind sie in diesem Fall auch keine verlässlichen Fake-News-Signale.

2.1.2 Merkmalextraktion

Bag-of-words

Das Bag-of-Words-Modell ist ein einfaches Verfahren zur Textrepräsentation, bei dem ein Dokument als Vektor der Häufigkeiten einzelner Wörter dargestellt wird – unabhängig von deren Reihenfolge oder Kontext. Es zählt lediglich das Vorkommen jedes Wortes aus einem festen Vokabular [17].

TF-IDF

TF-IDF ist ein gewichtetes Modell zur Textdarstellung, das berücksichtigt, wie häufig ein Wort in einem Dokument vorkommt (TF) und wie selten es im gesamten Kontext ist (IDF). Es dient dazu, häufige, aber wenig informative Wörter zu reduzieren und aussagekräftige Begriffe zu betonen [30].

Sparse Matrizen werden sowohl von Bag-of-Words als auch von TF-IDF genutzt. Eine Matrix wird als sparse bezeichnet, wenn der Anteil der Nicht-Null-Werte im Verhältnis zur Gesamtanzahl der Dokumente sehr klein ist. Pro hinzugefügtem Dokument wird eine Zeile erstellt und pro Wort im Vokabular eine Spalte. Da jedes Dokument nur einen Bruchteil der Wörter des Gesamtvokabulars enthält, bestehen der Großteil einer solchen Matrix aus Nullen.

	Doc 1	Doc 2	Doc 3
baking	0	1	1
cake	1	1	1
chocolate	1	0	0
he	0	0	1
her	0	0	1
is	0	1	1
loves	1	0	0
she	1	1	0
surprise	0	0	1
to	0	0	1

(a) Bag-of-words Sparse Matrix [12]

	Doc 1	Doc 2	Doc 3
baking	0	0.52	0.32
cake	0.34	0.40	0.25
chocolate	0.58	0	0
he	0	0	0.42
her	0	0	0.42
is	0	0.52	0.52
loves	0.55	0	0
she	0.44	0.52	0
surprise	0	0	0.42
to	0	0	0.42

(b) TF-IDF Sparse Matrix [12]

Abbildung 2.1: Vergleich der Sparse Matrizen

In Abbildung 2.1 wurden den beiden Matrizen jeweils die drei Dokumente:

- Doc1 - She loves chocolate cake
- Doc2 - She is baking a cake
- Doc3 - He is baking a cake to surprise her

hinzugefügt. In der Matrix 2.1a werden in jeder Zelle in welcher das Dokument das entsprechende Wort beinhaltet eine 1 gesetzt. In 2.1b wird statt einer 1 eine Gewichtung über die Häufigkeit der Wörter in allen Dokumenten hinweg erstellt und eingetragen. Sie bewertet die Wichtigkeit eines Wortes in einem Dokument relativ zur gesamten Sammlung von Dokumenten. Dabei wird die Termfrequenz (TF) mit der invertierten Dokumentfrequenz (IDF) multipliziert. Je höher der resultierende Wert, desto relevanter ist das Wort für das jeweilige Dokument. Die Formel lautet:

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D) \quad (2.1)$$

Dabei ist t das Wort, d das Dokument und D die gesamte Dokumentensammlung [6].

Die TF misst, wie häufig ein bestimmter Begriff t in einem Dokument d vorkommt. Sie beschreibt die lokale Bedeutung eines Wortes innerhalb des Dokuments.

$$\text{tf}(t, d) = \frac{\text{Anzahl der Vorkommen von } t \text{ in } d}{\text{Gesamtanzahl der Wörter in } d} \quad (2.2)$$

Die IDF bewertet, wie selten ein Begriff t in der gesamten Dokumentensammlung D ist. Je seltener ein Begriff in vielen Dokumenten vorkommt, desto höher ist sein IDF-Wert.

$$\text{idf}(t, D) = \log \left(\frac{N}{\text{df}(t)} \right) \quad (2.3)$$

Dabei ist N die Gesamtanzahl der Dokumente in der Matrix und $\text{df}(t)$ die Anzahl der Dokumente, in denen der Begriff t vorkommt [68].

Die in [27] beschriebene Relevance Frequency (RF) ist eine überwachte Gewichtungsform der IDF-Komponente im TF-IDF, die nicht nur zählt, in wie vielen Dokumenten ein Begriff vorkommt, sondern berücksichtigt, in welchen Klassen der Begriff besonders häufig oder exklusiv ist. Die Formel lautet:

$$rf(t) = \log \left(2 + \frac{P(t)}{\max(1, N(t))} \right) \quad (2.4)$$

Mit $P(t)$ für die Anzahl der relevanten Dokumente (z. B. positive Klasse), in denen der Term t und $N(t)$ für die Anzahl der irrelevanten Dokumente (z. B. negative Klasse), in denen der Term t vorkommt.

Während klassisches IDF ein Wort umso höher gewichtet, je seltener es allgemein in der Gesamtmatrix ist, gewichtet RF hingegen ein Wort umso höher, je stärker es mit einer bestimmten Zielklasse assoziiert ist. Dadurch hebt RF Begriffe hervor, die klassenunterscheidend sind was beim Arbeiten mit überwachten Modellen relevant ist.

In der IF-IDF wird für IDF wird nun also RF eingesetzt und es ergibt sich folgende Formel:

$$tfidf(t, d) = \frac{\text{Anzahl der Vorkommen von } t \text{ in } d}{\text{Gesamtanzahl der Wörter in } d} \cdot \log \left(2 + \frac{P(t)}{\max(1, N(t))} \right) \quad (2.5)$$

- Mit dem Wort t und dem Dokument d
- $P(t)$ für die Anzahl der relevanten Dokumente (z. B. positive Klasse), in denen der Term t vorkommt
- $N(t)$ für die Anzahl der irrelevanten Dokumente (z. B. negative Klasse), in denen der Term t vorkommt

Vergleich Bag-of-words und TF-IDF

Aus Tabelle 2.1 zu erkennen ist, dass TF-IDF in vielen Anwendungen leistungsfähiger ist als BoW. Insbesondere bei Texten mit hohem Vokabularumfang.

Hashing Vectorizer

Ein Hashing Vectorizer ist eine Methode zur Umwandlung von Text in numerische Merkmalsvektoren, ohne dass ein Vokabular explizit erstellt oder gespeichert wird. Stattdessen wird eine Hash-Funktion verwendet, um jedes Wort auf einen Index im Feature-Vektor abzubilden [12].

Bag-of-Words (BoW)	TF-IDF
Einfache Implementierung [17]	Berücksichtigt Wortwichtigkeit in gesamter Matrix [30]
Keine Gewichtung — häufige Wörter dominieren	Seltener, aber informativer Inhalt wird stärker gewichtet [21]
Hohe Dimensionalität in Sparse Matrix (jedes Wort bekommt eine separate Dimension) [20]	Gleiches Problem, aber mit informativeren Werten [4]
Ignoriert Wortreihenfolge und Kontext [82]	Gleiches Grundproblem, aber geringfügig bessere Performance [63]
Nützlich für einfache Klassifikatoren	Bessere Klassifikationsergebnisse in Kombination mit SVM oder Logistic Regression [44]

Tabelle 2.1: Vergleich der Vor- und Nachteile von BoW und TF-IDF

In Abbildung 2.2 wird der Vergleich zwischen Machine-Learning-Modellen unter Verwendung von BoW-, TF-IDF- und Hashing-Features gezeigt. Die y-Achse präsentiert den F1-Score, also die Precision und Recall-Werte der jeweiligen Modelle. Das Random-Forest-Modell (RF) zeigt eine schwache Leistung bei der Verwendung von Hashing, während die linearen Modelle (z.B. SVM (Support Vector Machine) und LR (Logistische Regression)) ihre F1-Werte mit Hashing-Features verbessern konnten.

Die Verbesserung ist aber nur minimal. Der F1-Score bei SVM ist ohne Hashing bei 0.89 und nach bei 0.90. Bei LR steigt der Wert auch von 0.87 und 0.88.

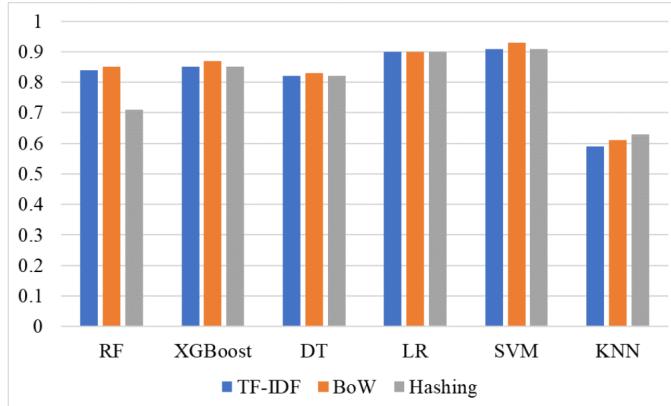


Abbildung 2.2: Vergleich verschiedener Modelle mit BoW, TF-IDF und Hashing [6]

2.1.3 Machine Learning Modelle

Naive-Bayes

Der Naive-Bayes-Algorithmus ist ein einfacher, aber leistungsfähiger Klassifikator, der auf dem Satz von Bayes basiert. Er wird häufig in Bereichen wie Textklassifikation, Spam-Erkennung und Sentiment-Analyse eingesetzt [92].

Der Klassifikator nutzt den Satz von Bayes zur Berechnung der Wahrscheinlichkeit einer Klasse C gegeben eine Merkmalsmenge X :

$$P(C | X) = \frac{P(X | C) \cdot P(C)}{P(X)} \quad (2.6)$$

Dabei ist:

- $P(C | X)$ – *Posterior*: Die Wahrscheinlichkeit für die Klasse C , nachdem die Daten X beobachtet wurden.
- $P(X | C)$ – *Likelihood*: Die Wahrscheinlichkeit, die Daten X zu sehen, wenn sie zur Klasse C gehören.
- $P(C)$ – *Prior*: Die ursprüngliche Wahrscheinlichkeit der Klasse C , ohne Kenntnis über die Daten.
- $P(X)$ – *Evidenz*: Die Gesamtwahrscheinlichkeit, die Daten X zu beobachten (über alle Klassen hinweg).

Die zentrale Annahme des Naive-Bayes-Klassifikators ist die bedingte Unabhängigkeit der Merkmale:

$$P(X | C) = \prod_{i=1}^n P(x_i | C) \quad (2.7)$$

Dies vereinfacht die Berechnung erheblich, da nur die Wahrscheinlichkeiten einzelner Merkmale betrachtet werden müssen [90].

Decision Tree

Ein Decision Tree (Entscheidungsbaum) ist ein Algorithmus für Klassifikation und Vorhersage. Er basiert auf einer baumartigen Struktur, bei der jeder Knoten bzw. Ast ein Merkmal aus einem Datensatz repräsentiert. Diese Struktur ermöglicht es, schrittweise Entscheidungen zu treffen, die schließlich zu einer Klassenzuordnung an einem Blattknoten führen [11].

Der Baum wird durch Auswahl von Merkmalen aufgebaut, die die Daten am besten aufspalten. Dieses Auswahlkriterium basiert auf dem Konzept der **Entropie** und dem daraus abgeleiteten **Informationsgewinn**. Ziel ist es, bei jeder Entscheidung im Baum das Merkmal auszuwählen, das die größte Reduktion an Unsicherheit bietet.

Die Entropie misst die Unreinheit oder Unbestimmtheit eines Datensatzes. Sie ist dann maximal, wenn alle Klassen gleichverteilt sind, und minimal (d.h. null), wenn alle Daten zur selben Klasse gehören. Die Entropie $E(S)$ eines Datensatzes S wird wie folgt berechnet:

$$E(S) = - \sum_{i=1}^c p_i \log_2 p_i \quad (2.8)$$

Dabei ist:

- c die Anzahl der Klassen,
- p_i der Anteil der Klasse i im Datensatz S .

Der Informationsgewinn misst die Reduktion der Entropie, die durch das Aufteilen eines Datensatzes mittels eines bestimmten Merkmals erzielt wird. Je größer der Informationsgewinn, desto besser ist das Merkmal für die Aufspaltung geeignet. Eine alternative Formel für den Informationsgewinn $IG(E)$ lautet:

$$IG(E) = 1 - \sum_{i=1}^c p_i^2 \quad (2.9)$$

Über einen Hyperparameter kann die maximale Tiefe des Baumes festgelegt werden. Eine zu große Tiefe kann zu Overfitting führen, da der Baum zu sehr an die Trainingsdaten angepasst wird [6].

Random Forest

Ein Random Forest besteht aus einer großen Anzahl von Entscheidungsbäumen. Jeder Baum wird auf einem zufällig gezogenen Teildatensatz trainiert (Bagging). Bei der Bildung jedes Knotens (Split) wird eine zufällige Teilmenge von Merkmalen berücksichtigt. Die finale Klassifikation ergibt sich durch Mehrheitsentscheidung aller Bäume (Ensemble Voting) [29].

Die Bedeutung eines Merkmals i im Random Forest ergibt sich aus der durchschnittlichen normierten Bedeutung dieses Merkmals über alle Entscheidungsbäume hinweg. Diese kann mathematisch wie folgt dargestellt werden:

$$RFf_i = \frac{\sum_{j \in \text{all trees}} norm f_{ij}}{T} \quad (2.10)$$

Dabei ist:

- RFf_i die Gesamtrelevanz der Klasse i im gesamten Wald,
- $norm f_{ij}$ die normierte Wichtigkeit des Merkmals i im Baum j ,
- T die Gesamtzahl der Entscheidungsbäume [6].

Wichtige Hyperparameter sind hier:

- `n_estimators`: Anzahl der Entscheidungsbäume im Wald.
- `max_depth`: Maximale Tiefe der Bäume.
- `max_features`: Anzahl der Merkmale, die für einen Split berücksichtigt werden.
- `bootstrap`: Gibt an, ob Stichproben mit Zurücklegen gezogen werden.

Im Vergleich zu Decision Trees ist Random Forest robuster gegenüber Overfitting und bringt durch das Ensemble Voting eine höhere Genauigkeit [3].

Support Vector Machines

Support Vector Machines (SVMs) sind überwachte Lernalgorithmen, die besonders effektiv für Klassifikationsaufgaben sind. Sie finden breite Anwendung in Bereichen wie Bioinformatik, Textklassifikation und insbesondere in der Erkennung von Fake News. Das Ziel einer SVM ist es, eine Trennlinie — oder in höherdimensionalen Räumen ein Trenn-Hyperplane — zu finden, das Datenpunkte verschiedener Klassen mit maximalem Abstand (Margin) voneinander trennt [60, 12, 71, 45].

1. Ziel: Eine Trennebene finden Eine SVM sucht eine Gerade (in 2D), Ebene (in 3D) oder Hyperplane, die die Klassen voneinander trennt:

$$w^T x + b = 0 \quad (2.11)$$

2. Bedingung für korrekte Trennung Für jeden Punkt x_i mit zugehörigem Label $y_i \in \{-1, +1\}$ gilt:

$$y_i(w^T x_i + b) \geq 1 \quad (2.12)$$

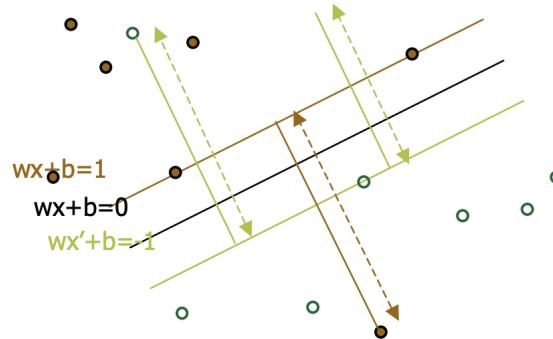


Abbildung 2.3: Darstellung von Hyperplanes [45]

3. Margin maximieren Der Margin ist der Abstand der nächsten Punkte beider Klassen zur Trennebene:

$$M = \frac{2}{\|w\|} \quad (2.13)$$

Daher wird zur Maximierung des Margins folgende Zielfunktion minimiert:

$$\min_{w,b} \quad \frac{1}{2} \|w\|^2 \quad \text{mit} \quad y_i(w^T x_i + b) \geq 1 \quad (2.14)$$

4. Fehler zulassen – Soft Margin

Bei nicht perfekt trennbaren Daten werden sogenannte Slack-Variablen ξ_i eingeführt:

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad (2.15)$$

Ziel ist es, Fehler klein zu halten, gleichzeitig aber den Margin möglichst groß:

$$\min \left(\frac{1}{2} \|w\|^2 + C \sum \xi_i \right) \quad (2.16)$$

5. Nichtlineare Trennung – Kernel-Trick

Bei komplexen Datensätzen wird das Problem durch eine Funktion ϕ in höhere Dimensionen überführt, ohne sie explizit zu berechnen [45]:

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \quad (2.17)$$

Gängige Kernel-Funktionen:

- Linearkernel: $K(x, x') = x^T x'$
- Polynomial: $K(x, x') = (x^T x' + 1)^d$
- Radial Basis Function (RBF): $K(x, x') = \exp(-\gamma \|x - x'\|^2)$ [45]

[60] zeigt, dass durch geeignete Wahl eines Kernels auch komplexe strukturierte Daten erfolgreich klassifiziert werden können.

Es ergibt sich folgender Ablauf:

1. Definiere Trennebene: $w^T x + b = 0$
2. Erzwinge korrekte Trennung: $y_i(w^T x_i + b) \geq 1$
3. Maximiere Margin: $\min \frac{1}{2} \|w\|^2$
4. Erlaube kleine Fehler: $\min \frac{1}{2} \|w\|^2 + C \sum \xi_i$
5. Wende ggf. Kernel an: $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$

Vorteile von SVMs sind:

- **Robustheit gegenüber Overfitting**, insbesondere bei hoher Dimensionalität und geringem Datensatz [60].
- **Gute Generalisierungsfähigkeit** durch Maximierung des Margins.
- **Effizient in der Praxis**: Für viele reale Probleme sind SVMs konkurrenzfähig gegenüber tieferen Netzwerken, (z.B. in Fake-News-Klassifikationen) [12, 71].
- **Flexibilität durch Kernel-Funktionen**, womit verschiedene Datentypen (z.B. Vektoren, Strings, Graphen) verarbeitet werden können.

Logistische Regression

Die logistische Regression (LR) ist ein weit verbreitetes Verfahren des überwachten maschinellen Lernens zur Klassifikation binärer und multiklassiger Zielvariablen. Sie wird eingesetzt, um die Wahrscheinlichkeit zu berechnen, mit der eine Beobachtung zu einer bestimmten Klasse gehört [29, 6, 80]. Im Gegensatz zur linearen Regression verwendet LR eine Aktivierungsfunktion, typischerweise die Sigmoidfunktion, um Ausgaben zwischen 0 und 1 abzubilden. Diese Werte stellen Wahrscheinlichkeiten dar und werden zur Vorhersage diskreter Zielwerte genutzt [6]. Die Tabelle 2.2 zeigt die Aktivierungsfunktionen, mit deren entsprechend benötigten Zielvariablen.

Typ	Aktivierungsfunktion	Typ der Zielvariable
Binäre logistische Regression	Logit (Sigmoid): $\frac{1}{1+e^{-z}}$	Binär (0/1)
Multinomiale logistische Regression	Softmax: $\frac{e^{z_k}}{\sum_j e^{z_j}}$	Kategorisch (mehrere Klassen)
Ordinal logistische Regression	Cumulative Logit, Probit, Cloglog	Geordnete Klassen
Probit-Modell	$\Phi(z)$ (Normalverteilung)	Binär (0/1), robust gegen Ausreißer

Tabelle 2.2: Übersicht von Aktivierungsfunktion in der logistischen Regression [46, 39, 1]

Das Ziel der logistischen Regression ist es, eine Funktion zu finden, die die Wahrscheinlichkeit P berechnet, dass ein Eingabewert X zur Klasse $y = 1$ gehört. Dies geschieht zum Beispiel mittels der Sigmoidfunktion [46]:

$$P = \frac{1}{1 + e^{-(a+bX)}} \quad (2.18)$$

Dabei sind:

- P : Wahrscheinlichkeit für Klasse 1 (Wert zwischen 0 und 1)
- X : Eingabewert (Merkmalsvariable)
- b : Gewicht des Merkmals
- a : **Bias** oder **Intercept**, also der Schnittpunkt mit der y-Achse, verschiebt die Entscheidungsgrenze. Ohne Bias würde die Entscheidungsgrenze immer durch den Ursprung laufen, was in der Praxis selten sinnvoll ist [36].

In [29] wurde der Intercept explizit deaktiviert, wodurch sich die Gleichung zu $P = 1/(1 + e^{-bX})$ vereinfacht.

Vorteile der logistischen Regression:

- **Einfachheit und Interpretierbarkeit**: LR-Modelle sind leicht verständlich und liefern direkt interpretierbare Wahrscheinlichkeiten [6].
- **Effizienz**: Sie sind schnell trainierbar und benötigen relativ geringe Rechenleistung [80].
- **Flexibilität**: LR lässt sich für binäre, multinomiale und ordinale Klassifikationsprobleme erweitern [80].
- **Breite Anwendbarkeit**: Sie wird in zahlreichen Bereichen eingesetzt, von der Medizin bis zur Textklassifikation [6, 29].

K-Nearest Neighbor

Der K-Nearest Neighbor (KNN) Algorithmus ist ein überwachter Lernalgorithmus, der unter anderem für Klassifikationsprobleme eingesetzt wird. Er gehört zur Gruppe der sogenannten lazy learners, da kein expliziter Trainingsprozess stattfindet. Stattdessen wird der Trainingsdatensatz während der Vorhersage verwendet [86].

KNN klassifiziert neue Instanzen auf Basis ihrer Ähnlichkeit mit bereits bekannten Beispielen. Dazu berechnet es die Distanz zwischen dem Testpunkt und allen Trainingspunkten. Anschließend werden die K ähnlichsten Punkte ausgewählt und die Vorhersage erfolgt bei Klassifikation durch Mehrheitsentscheidung [6].

Die Distanzmessung erfolgt in der Regel über die **euklidische Distanz**, mit der gemessen wird, wie weit zwei Punkte im Merkmalsraum voneinander entfernt sind. Die Formel lautet:

$$E_d = \sqrt{\sum_{i=1}^k (x_i - y_i)^2} \quad (2.19)$$

Dabei sind:

- x_i : der i -te Wert des zu klassifizierenden Punkts
- y_i : der entsprechende i -te Wert eines bekannten Trainingspunkts
- k : die Anzahl der Merkmale (Dimensionen), z.B. bei Textdaten die Länge des Vektors

Je kleiner der Wert E_d , desto ähnlicher sind sich die beiden Punkte.

Der KNN-Algorithmus verwendet mehrere wichtige Hyperparameter:

1. **n_neighbors**: Gibt an, wie viele der nächsten Trainingspunkte zur Klassifikation berücksichtigt werden. Der Wert (oft als K bezeichnet) sollte basierend auf den Eigenschaften des Datensatzes gewählt werden [86, 6].
2. **weights**: Bestimmt, ob allen Nachbarn das gleiche Gewicht gegeben wird oder ob näher gelegene Punkte stärker gewichtet werden [75].
3. **metric**: Die Distanzmetrik zur Berechnung der Ähnlichkeit. Standardmäßig wird die euklidische Distanz verwendet, möglich sind aber auch Manhattan, Minkowski oder andere Metriken [6]

Auch wenn KNN einfach und intuitiv zu implementieren ist und keinen Trainingsprozess benötigt, zeigt Abbildung 2.4, dass es im Vergleich mit Naive Bayes (NB), logistischer Regression (LG) und Support Vector Machines (SVM) ein vergleichsweise ineffektives Model für NLP ist.

Hier wurde KNN zur Klassifikation in einem Stimmungserkennungs-System für die Amtssprache Kambodschas eingesetzt. Dabei diente KNN dem Vergleich mit anderen Ansätzen wie SVM und wurde insbesondere hinsichtlich seiner Leistung bei der Klassifikation von

Approach	Accuracy	Precision	Recall	F1-Score
NB	73%	73%	73%	73%
K-NN	79%	75%	79%	77%
RF	81%	78%	81%	78%
SVM	85%	81%	85%	83%
BiLSTM	86%	84%	86%	84%

Abbildung 2.4: Auswertung verschiedener NLP Algorithmen [67]

Textdaten bewertet, bei denen die Reihenfolge, der Zeitpunkt oder der Verlauf über die Zeit eine wichtige Rolle für die Interpretation und Analyse spielt.

XGBoost

eXtreme Gradient Boosting (XGBoost) ist eine Implementierung von Gradient Boosting Decision Trees (GBDT). Beim XGBoost wird das Modell durch die Addition mehrerer Entscheidungsbäume aufgebaut, welche als schwache Lernalgorithmen (base learners) fungieren. Anders als bei Random Forests, bei denen Bäume unabhängig voneinander trainiert und aggregiert werden, lernen die Bäume in XGBoost aufeinander aufbauend (siehe Abbildung 2.5). Die Vorhersage für ein Beispiel ergibt sich aus der Summe der Ausgaben aller zuvor gelernten Bäume. Dadurch entsteht ein starkes Modell, das schrittweise durch Fehlerkorrektur verbessert wird [66, 15, 6].

Ein zentraler Vorteil von XGBoost ist die integrierte Regularisierung, mit der das Modell Overfitting vermeiden kann. Dabei werden zwei Arten von Regularisierung eingesetzt:

- **L1-Regularisierung:** Bestraft große Gewichtswerte, indem sie einige Gewichte auf Null setzt. Dadurch hilft sie, unwichtige Merkmale automatisch zu entfernen.
- **L2-Regularisierung:** Bestraft extreme Gewichtswerte, ohne sie komplett zu eliminieren. Dies führt zu stabileren Modellen mit kleinen, gleichmäßigen Gewichten.

¹<https://flower.ai/blog/2023-11-29-federated-xgboost-with-bagging-aggregation/>

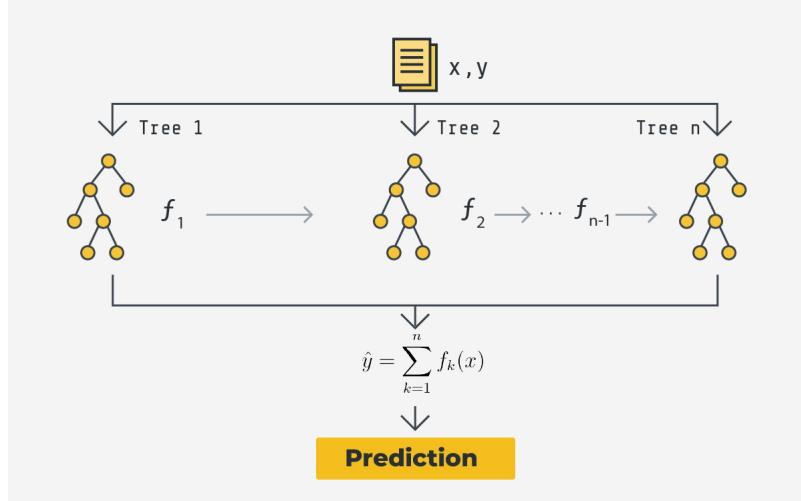


Abbildung 2.5: XGBoost¹

Beide Regularisierungen sind in die sogenannte Ziel- oder Kostenfunktion eingebettet, die das Modell bei jedem Trainingsschritt minimiert.

In Anwendungen der natürlichen Sprachverarbeitung (NLP) hilft diese Kombination, besonders bei großen Textmerkmalsräumen (z.B. TF-IDF), relevante Merkmale herauszufiltern und gleichzeitig stabile Modelle zu trainieren[15].

LightGBM

Das von [48] entwickelte Modell *Light Gradient-Boosting Machine (LightGBM)* ist eine hocheffiziente Implementierung eines GBDT. In Bezug auf dieses Modell wurden zwei zentrale Innovationen eingeführt, um das Training bei großen und hochdimensionalen Datensätzen zu beschleunigen, ohne die Modellgenauigkeit zu beeinträchtigen:

Gradient-based One-Side Sampling (GOSS) reduziert den Rechenaufwand von GBDT, indem es nur einen Teil der Dateninstanzen für die Berechnung der Informationsgewinne nutzt. Instanzen mit großen Gradienten (hohem Fehler) werden vollständig beibehalten, während aus den Instanzen mit kleinen Gradienten eine Stichprobe gezogen wird. Ein Gewichtungsschritt stellt sicher, dass die Verteilung der Daten korrekt bleibt. Dies führt zu einer deutlich schnelleren Trainingszeit bei nahezu gleichbleibender Genauigkeit.

Exclusive Feature Bundling (EFB) adressiert das Problem vieler hochdimensionaler Datensätze, in denen viele Merkmale nur selten (sogenannte *sparse* Features) oder nie gleichzeitig (*mutually exclusive*) aktiv sind. *Sparse* bedeutet, dass die meisten Werte in einem Merkmal Null sind, während *mutually exclusive* bedeutet, dass bestimmte Merkmale sich gegenseitig ausschließen – also nicht gleichzeitig einen von Null verschiedenen Wert annehmen. EFB fasst solche Merkmale zu sogenannten Bundles zusammen, wodurch sich die Anzahl der zu verarbeitenden Merkmale stark reduziert. Das Bündelungsproblem wird als Graphfärbungsproblem modelliert und mit einem Greedy-Algorithmus angenähert. Dadurch wird der Histogrammaufbau effizienter und das Training insgesamt beschleunigt.

Sowohl [48] als auch [43] zeigen, dass LightGBM gegenüber XGBoost effizienter trainiert, eine bessere Vorhersage gibt, weniger Merkmale benötigt und Kategorische Daten einfacher bearbeiten kann, dementsprechend also auch kein One-Hot-Encoding benötigt.

2.2 Deep Learning

2.2.1 Word Embeddings

Die klassischen Merkmalsextraktionen in Kapitel 2.1.2 eignen sich gut für klassische Machine Learning Modelle, wie Support Vector Machines oder Logistische Regression. Im Vergleich zu Word Embeddings erfassen diese aber keine semantischen Beziehungen. Word Embeddings verstehen die Bedeutung der einzelnen Wörter je nach Word Embedding in Teilen oder im gesamten Kontext [23] und repräsentieren dabei das ursprüngliche Wort in einem neuen Vektorraum, wobei aber die Eigenschaften des Wortes und seine Verbindungen zu anderen Wörtern bestmöglich bewahrt werden [73]. Dabei werden mit maschinellen Lerntechniken verschiedene dichte Vektoren mit einer festgelegten Dimension gebildet. Word Embeddings sind gegenüber zu BOW (Sparse Matrizen) deutlich speicherschonender.

Das Wort „Bank“ zum Beispiel hat in den Sätzen „Ich setze mich auf die Bank.“ und „Ich raube die Bank aus.“ zwei unterschiedliche Bedeutungen. Moderne Word Embeddings erkennt diese und erstellt für die zwei Kontexte/Wörter zwei verschiedene Vektoren [28].

In Abbildung 2.6 wird jedes Wort eines Korpus mit 6 Wörtern als dreidimensionaler Vektor dargestellt. Ziel von Word2Vec ist hierbei, dass Wörter mit ähnlichen Bedeutungen oder Kontexten ähnliche Vektordarstellungen haben. Die Ähnlichkeit der Vektoren „Katze“ und „Hund“ zeigt die semantische Beziehung zueinander. Die Vektoren „glücklich“ und „traurig“ hingegen zeigen in entgegengesetzte Richtungen, was auf ihre gegensätzlichen Bedeutungen hinweist [9].

Katze	[0.2, -0.4, 0.7]
Hund	[0.6, 0.1, 0.5]
Apfel	[0.8, -0.2, -0.3]
orange	[0.7, -0.1, -0.6]
glücklich	[-0.5, 0.9, 0.2]
traurig	[0.4, -0.7, -0.5]

Abbildung 2.6: Bsp. für Word Embeddings in einem dreidimensionalen Vektorraums [9]

Word2Vec

Das Word Embedding Word2vec verwendet ein neuronales Netzwerk und erfasst numerisch die Ähnlichkeiten zwischen Wörtern aufgrund ihrer kontextuellen Merkmale. Am häufigsten wird sie zur Analyse der semantischen Verbindungen zwischen Wörtern in einem Textkorpus eingesetzt [74].

Im Beispielsatz 'Mann verhält sich zu Frau wie König zu x.' erkennt Word2vec, dass für $x = \text{Königin}$ gilt. Word2Vec löst solche Aufgaben, indem es alle Wörter x' im Gesamtvokabular V ausprobiert und das Wort findet, das folgende Gleichung maximiert [16]:

$$\hat{x} = \underset{x' \in V}{\operatorname{argmax}} \sim(\vec{x'}, \vec{\text{king}} + \vec{\text{woman}} - \vec{\text{man}}) \quad (2.20)$$

Wie in Abbildung 2.7 zu sehen, gibt es für Word2Vec zwei verschiedene Implementierungen. Im CBOW-Modell (continuous bag-of-words) wird ein Wort aufgrund seines Kontextes vorhergesagt. Im Skip-gram-Modell wird hingegen Kontexte aufgrund eines Wortes vorhergesagt.

Bei einem relativ kleines Korpus, empfiehlt Google aufgrund seiner ausgeprägten Fähigkeit mit niedrigfrequenten Wörtern zu arbeiten, das Skip-gram-Modell anzuwenden [74].

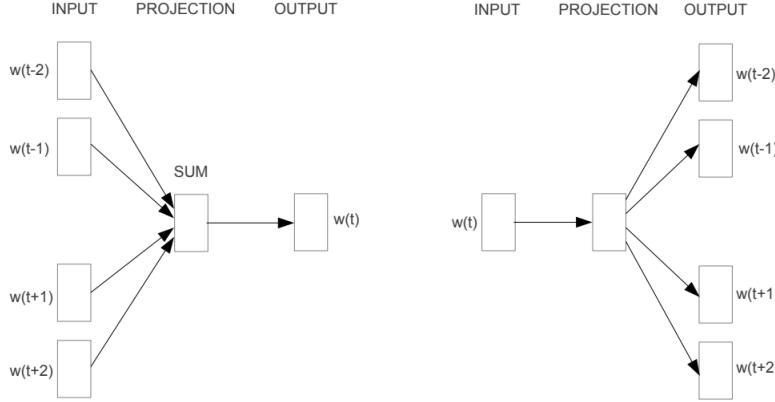


Abbildung 2.7: Vergleich CBOW (links) und Skip-gram (rechts) [57]

GloVe

Word2Vec fokussiert sich auf Informationen aus lokalen Kontextfenster, wobei globale Informationen hierbei nicht ausreichend genutzt werden. GloVe (Global Vectors for Word Representation) verwendet diese globalen Informationen, wodurch semantische Beziehungen zwischen Wörtern erfasst werden. Wie oft diese zusammen im Korpus vorkommen, wird in einer globalen Co-Occurrence-Matrix zusammengefasst [88].

Sei X eine Co-Occurrence-Matrix. Für jedes Wortpaar (i, j) zeigt X_{ij} , wie häufig das Wort w_j im Kontext von w_i erscheint.

Die bedingte Wahrscheinlichkeit, dass Wort j im Kontext von i erscheint, ist:

$$P(j | i) = \frac{X_{ij}}{X_i} \quad (2.21)$$

Zur Modellierung semantischer Beziehungen vergleicht GloVe Wahrscheinlichkeitsverhältnisse:

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k ice)/P(k steam)$	8.9	8.5×10^{-2}	1.36	0.96

Abbildung 2.8: Co-Occurrence-Wahrscheinlichkeiten für die Zielwörter „ice“ und „steam“ mit ausgewählten Kontextwörtern aus einem Korpus mit 6 Milliarden Tokens [65]

$$\frac{P_{ik}}{P_{jk}} = \frac{X_{ik}/X_i}{X_{jk}/X_j}$$

In Abbildung 2.8 zu erkennen ist, dass Werte > 1 gut mit Eigenschaften, die spezifisch für „ice“ sind und Werte < 1 gut mit Eigenschaften, die spezifisch für „steam“ sind korrelieren. Für $k = solid$ ist der Quotient 8.9. „solid“ hat somit eine größere semantische Beziehung mit „ice“ als mit „steam“. Für $k = gas$ ist der Wert 0.085. „gas“ passt folglich besser zu „steam“ als zu „ice“.

2.2.2 Deep Learning Modelle

CNN vs. RNN

Ein Convolutional Neural Network (CNN) ist ein Deep Learning Model (DNN) für Klassifikationsaufgaben, das Eingabedaten analysiert und dabei unterschiedlichen Merkmalen innerhalb der Daten Gewichtungen zuweist, um charakteristische Muster zu erkennen und verschiedene Klassen voneinander zu unterscheiden. Ein großer Vorteil von CNNs ist, dass sie wenig Datenvorverarbeitung benötigen, da sie Rohdaten direkt als Eingabe verarbeiten können [6].

Ein Recurrent Neural Network (RNN) ist ein DNN zur Verarbeitung sequentieller Daten. Im Vergleich zu CNNs können sich RNNs an frühere Eingaben erinnern, um aktuelle Vorhersagen zu beeinflussen [23]. RNN nutzt dabei den aktuellen Eingabewert sowie den vorherigen Ausgabewert in jedem Zeitschritt und trainiert sich damit selbst, indem es Fehler der Ausgabe zur Eingabe hinzu berechnet. RNN eignet sich somit besonders für Probleme in der natürlichen Sprachverarbeitung [88], da die Reihenfolge der Elemente in diesem Fall entscheidend ist.

Ein zentrales Problem bei RNNs ist jedoch das sogenannte Vanishing Gradient Problem, welches das Lernen langer Datenfolgen stark einschränken kann [6].

LSTM

Long Short-Term Memory (LSTM), ein RNN-Typ im Bereich der Sprachverarbeitung. Das Modell behebt das Problem des Vanishing Gradient Problems in klassischen RNNs, indem sie spezielle Speicherzellen verwenden, die Informationen über längere Zeiträume hinweg behalten können. Dadurch sind die LSTM-Modelle effektiv darin, langfristige Abhängigkeiten in sequenziellen Daten zu erfassen und Beziehungen zwischen Wörtern zu identifizieren [23].

Ein LSTM-Modell besteht aus mehreren Zellen, die nicht nur eine, sondern drei Aktivierungsfunktionen enthalten. Jede dieser Zellen speichert den Zustand des Problems über mehrere Zeitintervalle, während die drei Tore den Informationsfluss in die Zelle hinein und aus ihr heraus regulieren. Das Input-Gate, das Output-Gate und das Forget-Gate [10] (siehe Abbildung 2.9).

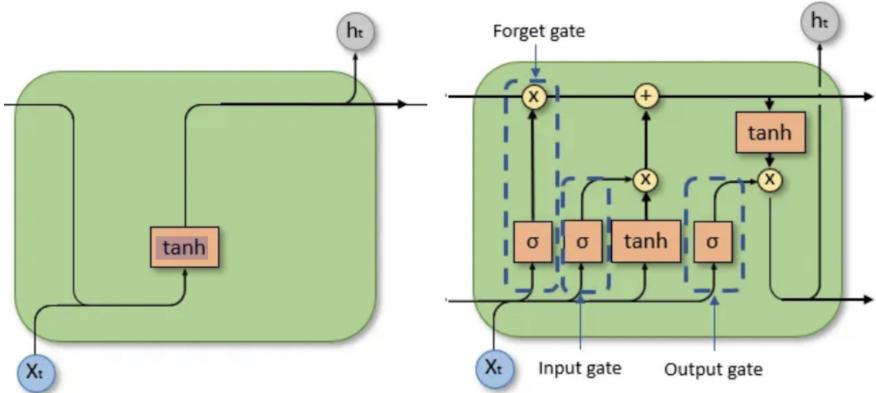


Abbildung 2.9: Vgl. RNN (links) und LSTM (rechts) [2]

Das Forget-Gate bestimmt, welche Informationen nicht mehr relevant sind und gelöscht werden können. Dies hilft, den Speicher der Zelle zu optimieren und unnötige Daten zu entfernen.

Das Input- und Output-Gate bestimmen, welche neuen Daten hinzugefügt und welche bestehenden Daten ausgegeben werden sollen. Sie arbeiten zusammen, um den Informationsfluss zu regulieren.

BiLSTM

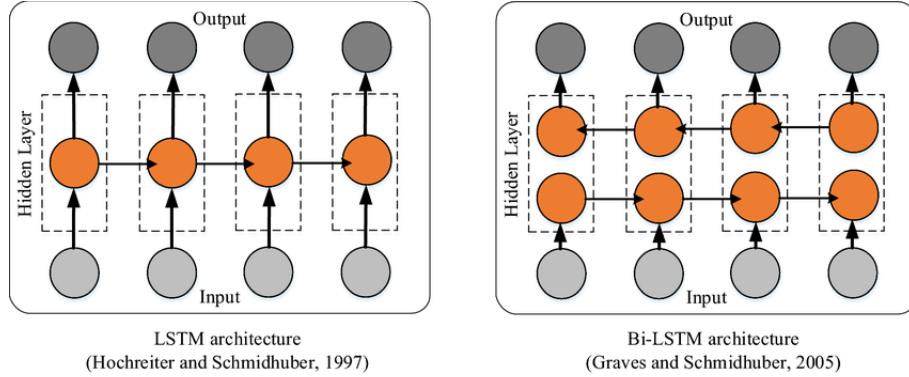


Abbildung 2.10: Vgl. LSTM und BiLSTM [78]

Bidirectional-LSTMs (BiLSTMs) sind eine Erweiterung der LSTM-Modelle, bei der zwei LSTMs auf die Eingabedaten angewendet werden. In der ersten Runde wird der Input von der ersten LSTM verarbeitet. Anschließend wird der Input in umgekehrter Form auf die zweite LSTM angewendet (siehe Abbildung 2.10). Der Input wird somit vor- und rückwärts gelesen, was das Erlernen von Langzeitabhängigkeiten verbessert und zu einer höheren Genauigkeit des Modells führt [58].

Der Hauptunterschied zwischen Bi-LSTMs und LSTMs besteht daher darin, dass Letztere nur Informationen aus der Vergangenheit bewahren, während in Bi-LSTMs durch die Kombination der beiden verborgenen Zustände sowohl Informationen aus der Vergangenheit als auch aus der Zukunft zu jedem Zeitpunkt erhalten bleiben können [78].

2.3 Transformer

2.3.1 Grundlagen der Transformer Architektur

Transformer sind erweiterte Deep-Learning-Modelle, welche sich aus einem Encoder und einem Decoder zusammensetzen (siehe Abbildung 2.11) und den sogenannten Self-Attention Mechanismus nutzen [85].

Im Vergleich zu RNNs, welche Kontexte nur von links nach rechts erkennen können (bzw. bi-direktional in BiLSTMs) kann der Kontext in Transformern global erkannt werden [37].

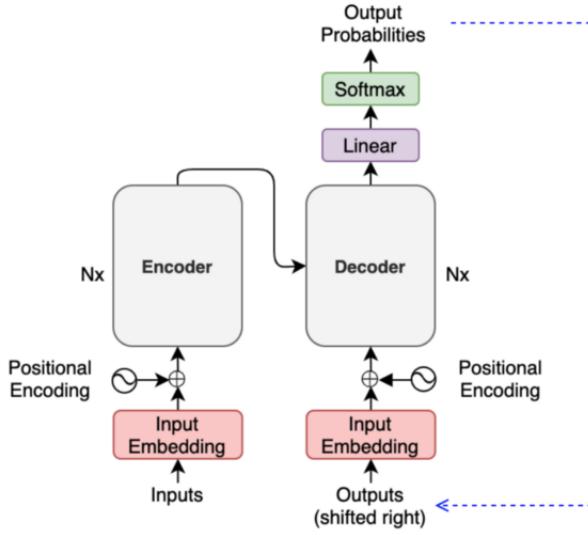


Abbildung 2.11: Eine Übersicht der Transformer Architektur [49] - vereinfacht von [85]

Self-Attention

In [85] wird die für Transformer entwickelte Self-Attention vorgestellt. Diese ermöglicht es, Beziehungen zwischen verschiedenen Positionen innerhalb einer einzigen Sequenz zu modellieren. Jede Position, also z.B. ein Wort in einem Satz, kann dabei auf alle anderen Positionen achten, um eine neue Darstellung der Sequenz zu erzeugen, die globale Abhängigkeiten widerspiegelt.

In dem Satz „Die Bank hat heute geschlossen.“ erkennt Self-Attention zum Beispiel, dass „Bank“ im Kontext von „geschlossen“ eher ein Gebäude und kein Möbelstück ist.

Folgende Schritte erklären, wie Self-Attention laut [85] funktioniert:

1. Einbettungen als Vektoren:

Jede Position der Eingabesequenz wird in einen Vektor eingebettet.

2. Erzeugung von Query, Key, Value (Q , K , V):

Aus diesen Vektoren werden durch lineare Transformationen die Matrizen Q (*Query*), K (*Key*) und V (*Value*) erzeugt.

3. Berechnung der Aufmerksamkeitsgewichte:

Die Self-Attention berechnet für jede Position die Kompatibilität zu allen anderen,

indem die Skalarprodukte von Q und K gebildet und durch $\sqrt{d_k}$ skaliert werden:

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

Diese Softmax-Gewichte bestimmen, wie stark eine Position auf andere achten soll.

4. Neues Repräsentationsvektor:

Die gewichteten Werte (V) werden aufsummiert und bilden so eine neue Darstellung für jede Position.

Statt nur eine Self-Attention zu berechnen, verwenden Transformer mehrere parallele *Heads*. Jeder *Head* lernt eine andere Perspektive auf die Sequenz. Die Ergebnisse aller *Heads* werden anschließend in einem *MultiHead* kombiniert:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (2.22)$$

Encoder-Decoder-Architektur

Ein Encoder-Decoder-Modell wird verwendet, um eine Eingabesequenz in eine Ausgabesequenz umzuwandeln. Zum Beispiel würde bei einer Textübersetzung der Encoder die Eingabesequenz sprachlich verstehen und der Decoder den übersetzten Text der Ausgabesequenz generieren.

Der Encoder nimmt eine Folge von Eingabewörtern und wandelt sie in eine Reihe von Vektoren um, die Informationen über jedes Wort und dessen Zusammenhang enthalten.

In der ursprünglichen Version der Encoder-Decoder-Architektur werden hierfür Recurrent Neural Networks (RNNs) oder bidirektionale RNNs verwendet. Diese lesen die Eingabe von vorne und hinten [7].

Im von [85] vorgestelltem Transformer-Modell werden RNNs durch Self-Attention ersetzt.

2.3.2 Hidden Layer

In einem Transformer-Modell [85] bestehen die *hidden layers* aus mehreren Encoder- und/oder Decoder-Schichten. Die erste Schicht erhält den Input und die letzte Schicht gibt den Output aus. Alle anderen Schichten sind versteckt und werden als *hidden layer* bezeichnet.

Im Falle eines Encoding Transformers wie BERT (siehe Kapitel 2.3.3) funktionieren *hidden layer* wie folgt [24]:

1. **Input-Einbettung:** Jeder Token wird in einen Zahlenvektor umgewandelt. Zusätzlich werden Positions- und Segmentinformationen hinzugefügt.
2. **Verarbeitung durch Hidden Layers:**
 - Jedes *hidden Layer* ist ein Transformer-Block mit Self-Attention.
 - Jeder Token wird an allen anderen Tokens angepasst, um seine Bedeutung im Kontext zu erfassen.
 - Diese Aufmerksamkeit ist bidirektional – sie bezieht sich auf Wörter davor und danach.
3. **Tiefe Verfeinerung:**
 - Frühe Schichten lernen einfache Beziehungen (z.B. Wortpaare, Syntax).
 - Spätere Schichten modellieren komplexere Abhängigkeiten (z.B. Bedeutung, logische Zusammenhänge).
4. **Ergebnis:**
 - Das letzte *hidden Layer* liefert eine kontext-sensitive Repräsentation für jeden Token.
 - In diesem Fall kann diese für Aufgaben wie Klassifikation, Fragebeantwortung oder Named Entity Recognition verwendet werden.

2.3.3 Transformer Modelle

BERT

Bidirectional Encoder Representations from Transformers (BERT) ist ein reiner, für Sprachverständnis optimierter, Encoder-Transformer [25]. Zwei wichtige Bestandteile der BERT Architektur sind die *Self Attention Heads* und die *Hidden Layers* (siehe Kapitel 2.3.1 & Abbildung A.1 und Kapitel 2.3.2 & Abbildung A.2).

Während CNNs und RNNs externe Word Embeddings wie Word2Vec oder GloVe verwenden, nutzt BERT eigene lernbare Embeddings.

Dazu noch einmal das Beispiel aus Kapitel 2.2.1: „Ich setze mich auf die Bank.“ und „Ich raube die Bank aus.“:

GloVe und Word2Vec erstellen einen festen Vektor für das Wort „Bank“, egal in welchem Satz es steht. Bei GloVe ist dieser Vektor ein Mittelwert aus allen Bedeutungen, die „Bank“ im Korpus je hatte. Der Vektor liegt folglich irgendwo zwischen Sitzmöbel und Finanzinstitut und repräsentiert keine der beiden Bedeutungen exakt.

BERT löst dieses Problem indem es kontextabhängige Embeddings erzeugt. So wird ein Vektor für das Wort „Bank“ erzeugt, der zur Bedeutung Sitzmöbel passt und ein weiterer für die Bedeutung Finanzinstitut.

Es verwendet während des Trainings Masked Language Modeling (MLM), um den Kontext und die Bedeutung von Wörtern im Satz zu verstehen. Anschließend wird es auf einem Datensatz mit gelabelten Bewertungen feinjustiert. Dabei verbindet es jedes Eingabeelement mit jedem Ausgabeelement und weist dabei wichtigen Wörtern und Phrasen im Text höhere Gewichtungen zu [23].

Im MLM wird ein bestimmter Teil der Wörter in der Eingabesequenz zufällig maskiert (siehe Abbildung 2.12), und das Modell muss diese verdeckten Wörter korrekt vorhersagen.

BERT nutzt die bi-direktionale Transformer-Architektur (siehe Abbildung 2.13), bei welcher tiefe semantische Informationen eines Satzes erfasst werden können. Aufgrund dieser Bidirektionalität ist das Modell bei späteren Vorhersagen effektiver [91]. [25] zeigt, wie relevant bidirektionale Pretrainings für qualitativ hochwertige Sprachrepräsentationen sind.

<i>m</i>	Example	PPL
15%	We study high [] ing rates [] pre-training language models .	17.7
40%	We study high [] rates [] pre- [] models .	69.4
80%	We [] high [] [] [] models []	1141.4

Random initialization

Abbildung 2.12: Bsp. zum MLM [91]

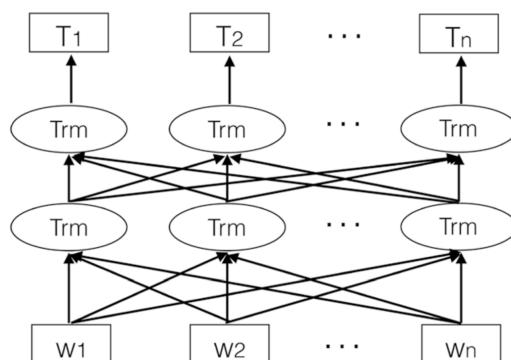


Abbildung 2.13: Bidirektonalität des BERT Modells [88]

Für das Erstellen der Wortvektoren nutzt BERT WordPiece Embeddings. WordPiece ist ein Tokenisierungsverfahren, das Wörter in kleinere Einheiten zerlegt. Es wurde von Google speziell für BERT entwickelt, um auch seltene oder unbekannte Wörter sinnvoll verarbeiten zu können.

Folgendes Beispiel für ein WordPiece Embedding (aus [32]):

1. Das **Startvokabular** besteht aus einem Vokabular aus Einzelbuchstaben (z.B. h, #e, #l, #o für "hello"), wobei alle Buchstaben außer dem Ersten mit # markiert werden, um zu zeigen, dass sie nicht am Wortanfang stehen.
2. **Häufigkeitsanalyse:** Identifiziert häufig gemeinsam auftretende Buchstabenpaare, z.B. ("##g", "#s") in "hugs".
3. **Mergeregeln:** Zum Zusammenfügen berechnet WordPiece einen Score:

$$\text{Score} = \frac{\text{Häufigkeit des Paars}}{\text{Häufigkeit Teil 1} \times \text{Häufigkeit Teil 2}} \quad (2.23)$$

Dadurch werden eher seltene Kombinationen zusammengefügt, die besser charakteristische Subwörter ergeben.

4. **Merge-Iterationen:** Das Zusammenfügen wird so lange wiederholt, bis das gewünschte Vokabular erreicht ist.

Beim Zerlegen neuer Wörter:

1. Suche das längste Subwort im Vokabular, das am Wortanfang passt.
2. Markiere alles danach mit # und wiederhole.
3. Wenn gar kein Teil im Vokabular ist, kommt das Sondertoken [UNK] (unbekannt) zum Einsatz.

Beispiele:

- "hugs" → ["hug", "#s"]
- "bugs" → ["b", "#u", "#gs"]
- "mug" → [UNK], falls #m nicht im Vokabular ist

Zusätzlich werden Positions- und Segment-Embeddings hinzugefügt (siehe Abbildung 2.14).

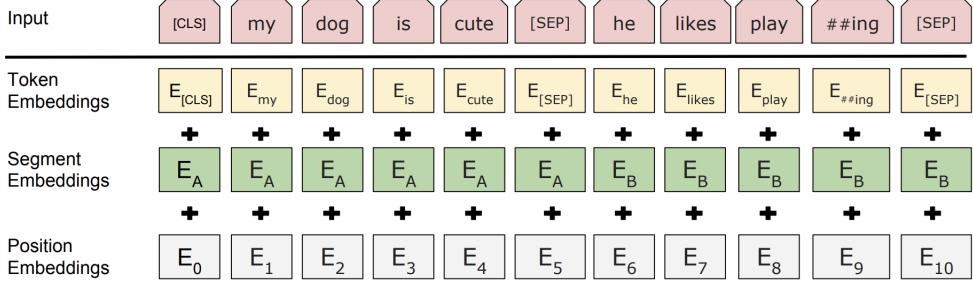


Abbildung 2.14: Zusammensetzung eines Input Tokens im BERT Modell [25]

Das Input Token ergibt sich aus dem Token-, Position- und Segment-Embedding. Das Position-Embedding (Position Encoding in Abbildung 2.11) stellt hierbei die jeweilige Position im Satz dar und das Segment-Embedding ordnet den Token dem entsprechenden Satz zu.

Vortrainiert wurde das BERT-Modell auf dem BookCorpus, einem Datensatz bestehend aus 11.038 unveröffentlichten Büchern, sowie auf der englischsprachigen Wikipedia (ohne Listen, Tabellen und Überschriften) [24].

RoBERTa

Aufgrund der Annahme, dass das in Kapitel 2.3.3 beschriebene BERT Modell nicht ausreichend trainiert wurde, entwickelte [54] das RoBERTa Modell.

Statt WordPiece nutzt RoBERTa Byte-Pair-Encoding (BPE) zum Tokenisieren wobei ein Vokabular von 50.265 Token verwendet wird.

BPE besteht aus folgenden, in [76] erklärten, Schritten:

1. Zerlege jedes Wort in einzelne Zeichen und hänge ein Endsymbol „.“ an (z. B. „lower“ → [l, o, w, e, r, .]).
2. Zähle alle benachbarten Zeichenpaare im Korpus (z. B. (l, o), (o, w), (w, e), ...).

3. Führe das häufigste Paar zu einem neuen Symbol zusammen (z.B. $(l, o) \rightarrow "lo"$).
4. Wiederhole Schritt 2–3 für eine feste Anzahl von Merges (z.B. $"lo" + "w" \rightarrow "low" \rightarrow [low, e, r, ·]$).
5. Speichere die entstandenen Merge-Regeln und wende sie auf neue Wörter an.

Die Eingaben bestehen aus Abschnitten von 512 aufeinanderfolgenden Token, die sich auch über mehrere Dokumente erstrecken können. Der Beginn und das Ende eines Dokuments werden durch spezielle Markierungen $< s >$ und $< /s >$ gekennzeichnet. Während des Pretrainings werden 15% der Token maskiert – in 80% der Fälle durch $<mask>$, in 10% durch ein zufälliges anderes Token und in den restlichen 10% bleiben sie unverändert. Anders als bei BERT erfolgt die Maskierung dynamisch, also in jeder Epoche neu.

Das RoBERTa-Modell wurde auf einer Zusammenführung von fünf Datensätzen vortrainiert: dem BookCorpus mit über 11.000 unveröffentlichten Büchern, der englischen Wikipedia (ohne Listen, Tabellen und Überschriften), CC-News mit 63 Millionen englischsprachigen Nachrichtenartikeln aus dem Zeitraum September 2016 bis Februar 2019, OpenWebText als Open-Source-Rekonstruktion des WebText-Datensatzes von GPT-2 sowie dem Stories-Datensatz, einem nach erzählerischem Stil gefilterten Ausschnitt aus CommonCrawl-Daten. Insgesamt umfassen diese Datensätze 160 GB an Text.

XML-RoBERTa

Im Gegensatz zu RoBERTa, das ausschließlich auf englischen Texten trainiert wurde, ist XLM-RoBERTa ein mehrsprachiges Transformer-basiertes Sprachmodell, das auf 100 Sprachen trainiert wurde. Die Trainingsdaten umfassen über 2 Terabyte gefilterter CommonCrawl-Texte. Auch die Vokabulargröße ist mit 250.002 Token deutlich größer als die 50.265 Token bei RoBERTa [19].

Statt Byte-Pair-Encoding (BPE) wie bei RoBERTa verwendet XLM-RoBERTa den SentencePiece Tokenizer. Dieser arbeitet sprachunabhängig, benötigt keine vorab segmentierten Daten und funktioniert auch auf rohem Unicode-Text. Besonders wichtig ist das Konzept der lossless Tokenization, das Segmentierung vollständig reversibel macht [51]. Sentence Piece setzt sich aus vier Hauptkomponenten zusammen. Dem Normalizer, Trainer, Encoder und Decoder.

1. Der **Normalizer** wandelt unter anderem alle Zeichen in ASCII um, normalisiert Leerzeichen und ersetzt alle Groß- mit Kleinbuchstaben.
2. Der **Trainer** erstellt aus diesen normalisierten Daten daraufhin ein Subwort-Modell. Jedes Subwort bekommt eine eindeutige ID. <s> markiert den Start eines Satzes und hat immer die ID 0. Das _ steht für ein Leerzeichen und markiert somit den Wortanfang.
3. Der **Encoder** kodiert nun einen übergebenen Input basierend auf dem erstellten Subwort-Modell in eine ID-Sequenz. item Der **Decoder** dekodiert eine erstellte ID-Sequenz zurück in einen lesbaren Input.

In Kapitel 2.3.3 wurde bereits die Funktionsweise des MLMs erklärt. Die Erweiterung von MLM ist das Translation Language Model (TLM). Im TLM werden statt einzelner Sätze (wie im MLM) übersetzte Satzpaare verwendet. Wörter in beiden Sprachen werden maskiert, und das Modell lernt, sie mithilfe beider Sprachversionen zu erraten (siehe Abbildung 2.15). So lernt es, die Bedeutungen über Sprachgrenzen hinweg zu verbinden.

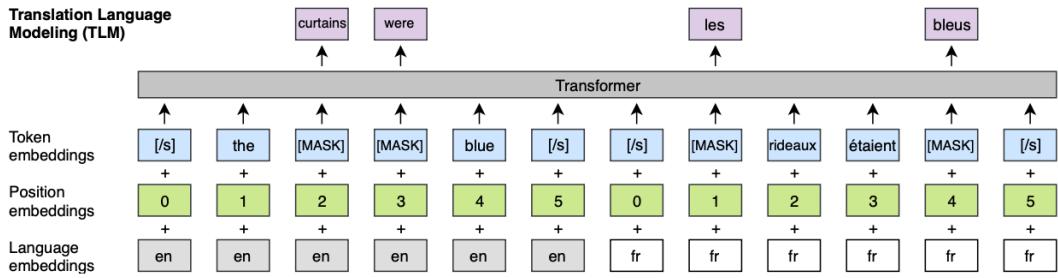


Abbildung 2.15: Auswertung eines bilingualen Satzpaars von TLM [18]

XLM ist ein mehrsprachiges Sprachmodell, das darauf ausgelegt ist, Sprachverständnis über mehrere Sprachen hinweg zu ermöglichen. Votrainiert wird es mit MLM und zusätzlich TLM. Durch diese Kombination kann das Modell lernen, ähnliche Inhalte in verschiedenen Sprachen miteinander zu verknüpfen [18].

2.4 Metriken

Zur Auswertung der überwachten Modelle werden Metriken genutzt. Die Auswahl der richtigen Metriken hängt von der gewünschten Zielsetzung ab. Diese kann zum Beispiel binäre, bzw. multi-Klassen Klassifikation oder Regression sein.

Fake News Erkennung ist eine binäre Klassifizierung (Der Artikel ist entweder 'wahr' oder 'falsch').

Dabei ergeben sich vier mögliche Ausgänge bei der Modellvorhersage:

- True Positive (TP) - Das Modell hat die positive Klasse vorhergesagt.
(Der Artikel, der kein Fake ist, wird als 'wahr' gedeutet)
- True Negative (TN) - Das Modell hat die negative Klasse richtig vorhergesagt.
(Der Artikel, der Fake ist, wird als 'falsch' gedeutet)
- Falsch positiv (FP) - Das Modell hat die positive Klasse falsch vorhergesagt.
(Der Artikel, der kein Fake ist, wird als 'falsch' gedeutet)
- Falsches Negativ (FN) - Dein Modell hat die negative Klasse falsch vorhergesagt.
(Der Artikel, der kein Fake ist, wird als 'wahr' gedeutet)

Diese vier Werte werden in einer sogenannten Konfusionsmatrix (siehe Abbildung 2.16) zusammengefasst aus der verschiedene Bewertungsmaßen abgeleitet werden können.

Nach [50, 70] sind die relevantesten Metriken für binäre Klassifikationen Accuracy, Recall (Sensitivity in [70]), Specificity und Precision.

2.4.1 Accuracy

Die Accuracy gibt den Anteil korrekt klassifizierter Instanzen an.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.24)$$

		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

Abbildung 2.16: Konfusionsmatrix [22]

2.4.2 Recall

Der Recall gibt den Anteil korrekt erkannter positiver Fälle an.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.25)$$

2.4.3 Specificity

Die Specificity gibt den Anteil korrekt erkannter negativer Fälle an.

$$\text{Specificity} = \frac{TN}{TN + FP} \quad (2.26)$$

2.4.4 Precision

Die Präzision gibt den Anteil tatsächlich positiver Fälle unter allen als positiv vorhergesagten Fällen an.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2.27)$$

2.4.5 F1-Score

Der F1-Score vereint die beiden Metriken Recall und Precision in einem einzigen Wert und ist hilfreich, wenn ein Gleichgewicht zwischen diesen beiden wichtig ist – vor allem bei unausgeglichenen Datensätzen, bei denen Accuracy allein irreführend sein kann.

Sind in dem Datensatz der Fake News Erkennung zum Beispiel 95% der Artikel 'wahr' und 5% 'falsch' hat ein Modell das ausschließlich 'wahr' vorhersagt eine Accuracy von 95%. Es erkennt aber keinen einzigen Artikel der Fake ist. Der Recall wäre in diesem Fall 0 und somit auch der F1-Score.

$$\text{F1-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.28)$$

2.5 Hybride Modelle zur Fake News Erkennung

2.5.1 CNN und LSTM mit PCA

In [83] wird eine hybride neuronale Netzwerkarchitektur vorgeschlagen, die die Fähigkeiten von CNN und LSTM kombiniert. Dabei kommen zwei unterschiedliche Verfahren zur Dimensionsreduktion zum Einsatz: Hauptkomponentenanalyse (PCA) und Chi-Quadrat-Verfahren. Diese Verfahren werden empfohlen, um die Dimensionalität der Merkmalsvektoren zu verringern, bevor diese an den Klassifikator weitergeleitet werden.

Model	Accuracy	Precision	Recall	F-score
CNN-LSTM without pre-preprocessing	78.4%	81.4%	82.4%	81.9%
CNN-LSTM with pre-preprocessing	93%	96%	97%	96%
CNN-LSTM with Chi-Square	95.2%	92.3%	91.1%	91.49%
CNN-LSTM with PCA	97.8%	97.4%	98.2%	97.8%

Abbildung 2.17: Ergebnisse der Klassifizierungen der verschiedenen CNN-LSTM Modelle [83]

Wie in Abbildung 2.17 zu erkennen, werden verschiedenen Modelle miteinander verglichen. Einmal mit und ohne Vorverarbeitung der Daten und dann mit den verschiedenen Dimensionsreduktionen PCA und Chi-Quadrat. PCA schneidet mit einem F1-Score von 97,8% am Besten ab, während keine Vorverarbeitung der Daten nur einen F1-Score von 81,9% erreicht.

Die Modelle wurden auf Basis des FNC-1 Datensatzes gemessen, in welchem 2.587 Artikeln etwa 300 verschiedenen Schlagzeilen zugeordnet werden sollen. Jeder Artikel wird in Bezug auf eine Schlagzeile einer von vier Klassen zugeordnet:

- **Agree:** Artikel stimmt der Schlagzeile zu
- **Disagree:** Artikel widerspricht der Schlagzeile
- **Discuss:** Artikel diskutiert das Thema der Schlagzeile
- **Unrelated:** Artikel ist thematisch nicht verwandt mit der Schlagzeile

Zum Vergleich wurden weitere Modelle wie BERT auf den Datensatz angewendet. Dieses erzielte eine Accuracy von 91,3% während das hybride CNN-LSTM Modell eine Accuracy von 97,8% erreichte.

2.5.2 CNN und LSTM mit GloVe

Ein weiteres in [12] vorgestelltes hybrides Modell ist zusammengesetzt aus CNN und LSTM. Zusätzlich werden die Daten mit dem GloVe Embedding vorverarbeitet.

	Manual Dataset Accuracy (%)		Extended Dataset Accuracy (%)	
Model	Train	Test	Train	Test
CNN w/ LSTM	92.3	85.6	93.3	91.1
GRU	95.1	84.3	92.2	90.0
LSTM	91.5	82.1	92.8	87.8

Abbildung 2.18: Ergebnisse verschiedener Modelle mit GloVe Embedding [12]

Getestet wurden die Modelle CNN mit LSTM, GRU und LSTM. Genutzt wurden zwei verschiedene Datensätze. Ein manuell erzeugter aus über 1500 Quellen basierend auf ca. 9000 Artikeln mit Falschinhälften und ca. 9000 echten Artikeln und ein erweiterter, welcher zusätzlich mit Inhalten aus anderen Datensätzen von Kaggle oder GitHub bereichert wurde. Durch die zusätzliche Nutzung von GloVe Embeddings wird in dieser Arbeit das Overfitting reduziert (siehe Abbildung 2.18). Vergleichsweise ist die Accuracy beim erweiterten Datensatz mit dem Keras Embedding bei 98,4% im Training und bei 89,5% beim Test (CNN mit LSTM).

[12] stellt außerdem fest, dass Wort-Embeddings wie GloVe die Semantik des Textes deutlich besser erfassen als Techniken zur Merkmalsextraktion wie Bag-of-Words oder TF-IDF. In Kombination mit Deep-Learning-Modellen liefern sie eine höhere Genauigkeit als herkömmliche Machine-Learning-Modelle.

2.5.3 BERT und CNN (FakeBERT)

Ein von [47] vorgestelltes Modell trägt den Namen *FakeBERT* und setzt sich aus den Modellen BERT und CNN zusammen. BERT analysiert den Text und versteht den Zusammenhang der Wörter, während mehrere kleine CNNs gleichzeitig verschiedene Merkmale aus dem Text herausfiltern. Die Ergebnisse dieser CNNs werden zusammengeführt, weiterverarbeitet und klassifiziert. Entschieden wird auch in diesem Modell ob es sich um echte oder falsche Nachrichten handelt. Das Modell erkennt durch diese Architektur sowohl den allgemeinen Sinn als auch feine sprachliche Muster im Text.

Gearbeitet wurde mit dem *fake-and-real-news-dataset* von Kaggle. Dieser besteht aus 20.800 Nachrichtenartikeln, die während der US-Präsidentenwahl 2016 gesammelt wurden. Enthalten sind unter anderem Merkmale wie Titel, Autor, Textinhalt. Klassifiziert werden die Nachrichten als echt oder gefälscht. Im Datensatz gibt es 10.540 echte und 10.260 gefälschte Artikel.

[47] zeigt, dass das vorgeschlagene Modell FakeBERT mit einer Genauigkeit von 98,90% (siehe Abbildung 2.19) deutlich besser abschneidet als klassische Machine-Learning-Modelle und bis 2021 bestehende Deep-Learning-Ansätze. Durch die Kombination von BERT als kontextuelles Sprachmodell mit mehreren parallel laufenden CNN-Blöcken zur Merkmalsextraktion können sowohl globale Bedeutungszusammenhänge als auch lokale sprachliche Muster zuverlässig erfasst werden.

2.5.4 BERT und CNN (MCred)

Wie auch in Kapitel 2.5.3 entwickelt [87] in seiner Arbeit ein weiteres hybrides Modell aus BERT und CNN. Dieses trägt den Namen *MCred*.

Es wird auf vier verschiedenen Datensätzen trainiert:

- **WELFake:** Enthält 37.106 gefälschte und 35.028 echte Nachrichten und dient als Hauptdatensatz für das MCred-Modell.

Word embedding model	Classification model	Accuracy (%)
TF-IDF (using unigrams and bigrams)	Neural Network	94.31
BOW (Bag of words)	Neural Network	89.23
Word2Vec	Neural Network	75.67
GloVe	MNB	89.97
GloVe	DT	73.65
GloVe	RF	71.34
GloVe	KNN	53.75
BERT	MNB	91.20
BERT	DT	79.25
BERT	RF	76.40
BERT	KNN	59.10
GloVe	CNN	91.50
GloVe	LSTM	97.25
BERT	CNN	92.70
BERT	LSTM	97.55
BERT	Our Proposed model (FakeBERT)	98.90

Abbildung 2.19: Ergebnisse der verschiedenen Modelle bei Validierung [47]

- **Kaggle Fake News Dataset:** Umfasst 10.369 gefälschten und 10.349 echte Nachrichten mit den Merkmalen Titel, Text und Autor.
- **McIntire Dataset:** Besteht aus 3.164 gefälschten und 3.171 echten Nachrichtenartikeln zur US-Präsidentenwahl 2016.
- **FakeNewsNet:** Enthält 24.396 gefälschte und 13.614 echte Nachrichten. Stammt aus diversen Quellen und deckt zahlreiche Themenfelder ab.

FakeBERT kombiniert BERT direkt mit parallelen CNNs zur Merkmalsextraktion, während MCred BERT und CNN getrennt verarbeitet und ihre Ausgaben später zusammenführt. Zudem nutzt MCred zusätzlich GloVe-Embeddings, was FakeBERT nicht tut. Wie in Abbildung 2.20 zu erkennen erreicht MCred dadurch im Kaggle Datensatz mit einer Accuracy von 99,46% einen besseren Wert als FakeBERT (Rohit Kumar Kaliyar and Narang (2021)) mit 98,90%.

In [26] wurde MCred im August 2024 mit einer Accuracy von 99,01% als zweitbeste State-of-the-Art-Technik benannt.

	Mersinias et al. (2020)	Khan and Alhazmi (2020)	Kaliyar et al. (2020)	Rohit Kumar Kaliyar and Narang (2021)	MCred
Dataset accuracy	Kaggle: 97.52% McIntire: 94.53% Fakenews: 96.78%	Kaggle: 90.70%	Kaggle: 98.36%	Kaggle: 98.90%	Kaggle: 99.46% McIntire: 97.16% FakeNews: 97.98% WELFake: 99.01%
Document representation features	Class label frequency distance vector	Doc2Vec	GloVe	BERT embeddings	GloVe – BERT embeddings
Classifier	Logistic regression (ML) CNN + LSTM (DL)	AdaBoost LinearSVM	Deep CNN	CNN	CNN, BERT

Abbildung 2.20: Ergebnisse der verschiedenen Modelle [87]

2.5.5 BERT und LSTM

[69] schlägt ein hybrides Modell vor, welches BERT und LSTM kombiniert. Hierbei fungiert BERT nachdem die Daten bereinigt wurden als Tokenizer und Basismodell durch seine Fähigkeit zur tiefen und kontextuellen Wortrepräsentationen (siehe Abbildung 2.21). Anschließend werden die erzeugten Embeddings in ein LSTM gegeben, in welchem dessen Zellen die Langzeitabhängigkeiten aus den Sequenzen speichern.

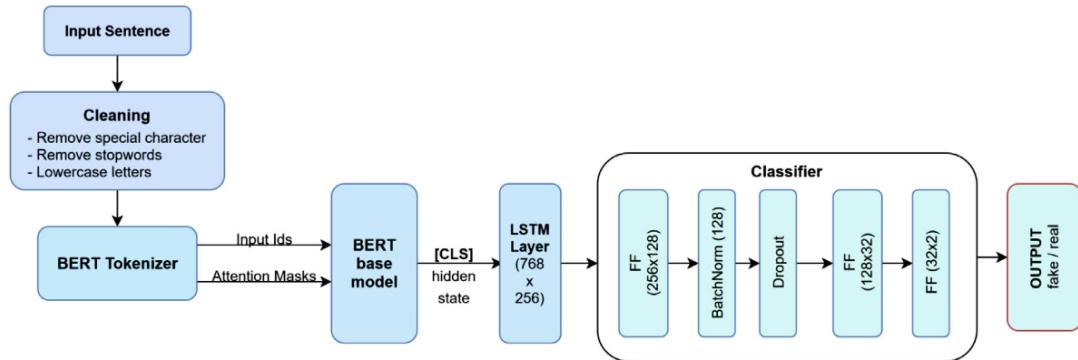


Abbildung 2.21: Architektur des vorgeschlagenen hybriden Modells [69]

Ziel des Papers ist es, die semantische Tiefe von BERT mit der temporalen Lernfähigkeit von LSTM zu verbinden, um Fake-News besser erkennen zu können.

Für die Evaluation wurde der FakeNewsNet-Datensatz verwendet. Dieser besteht aus dem PolitiFact- und dem GossipCop-Datensatz. PolitiFact enthält politisch orientierte Nachrichten, während GossipCop Nachrichten aus dem Unterhaltungsbereich beinhaltet.

In beiden Datensätzen sind die Inhalte kategorisiert in echte und gefälschte Nachrichten. Die Datensätze bestehen aus Nachrichtentiteln. PolitiFact umfasst 432 falsche und 624 echte Titel, GossipCop 5323 falsche und 16.817 echte Titel.

	Accuracy (%)	Precision	Recall	F1 Score
TCNN-URG	71.20	0.71	0.94	0.81
LIWC	76.90	0.84	0.79	0.81
CSI	82.70	0.84	0.89	0.87
HAN	83.70	0.82	0.89	0.86
SAFE (Multimodal)	87.40	0.88	0.90	0.89
BERT	86.25	0.90	0.87	0.88
BERT + LSTM	88.75	0.91	0.90	0.90

Abbildung 2.22: Ergebnisse der verschiedenen Modelle mit dem PolitiFact Datensatz [69]

Das vorgeschlagene Modell, übertrifft klassische Methoden und auch reines BERT in der Fake-News-Erkennung. Auf dem PolitiFact-Datensatz erreicht es 88,75% Genauigkeit (siehe Abbildung 2.22, bei reinem BERT sind es nur 86,25%. Der LSTM-Layer verbessert dabei die Nutzung der kontextuellen BERT-Embeddings, insbesondere bei der Analyse sprachlicher Muster in Newstiteln.

2.5.6 BERT und BiLSTM

[89] stellt ein hybrides BERT und BiLSTM Modell vor. Hierbei wird ein vortrainiertes BERT-Modell als Merkmalsencoder genutzt. Dabei bleiben alle internen Gewichte und Parameter von BERT unverändert und werden nicht weiter trainiert. Zusätzlich wird ein BiLSTM Modell für die weitere Verarbeitung der BERT-Ausgaben trainiert (siehe Abbildung 2.23).

Verwendet wurde ein COVID-19 Fake News Dataset von Kaggle, bestehend aus 6.420 Trainings- und 2.140 Testeinträgen. Jeder Eintrag enthält die Merkmale Tweet-ID, den Text des Tweets und ein Label (echt oder gefälscht). Augrund der Größe dieses Datensatzes werden die internen Gewichte und Parameter von BERT nicht weiter trainiert. Durch die etwa 8.500 Einträge ist der COVID-19-Datensatz zu klein, um das BERT-Modell effektiv und ohne Overfitting neu zu trainieren. Durch das Einfrieren wird verhindert, dass die vortrainierten Sprachmuster überschrieben werden. So bleibt die allgemeine Sprach-

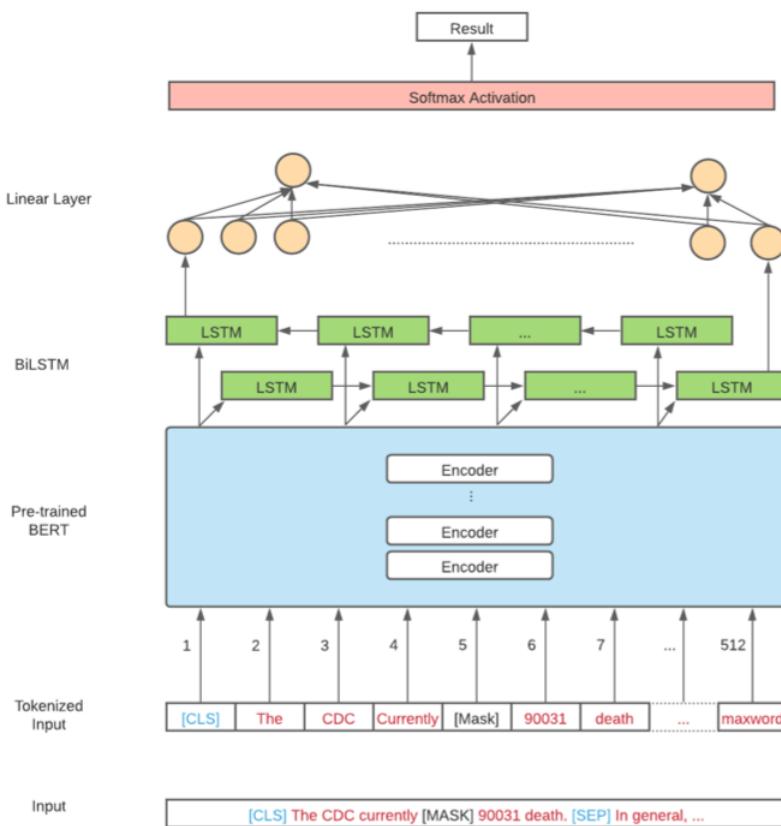


Abbildung 2.23: Architektur des hybriden Modells [89]

kompetenz von BERT erhalten, während das darauf aufbauende BiLSTM lernt, diese Informationen für die Fake-News-Erkennung zu nutzen.

Verglichen wurde von [89] folgende Modelle:

- **Modell 1:** Feinjustiertes BERT-Modell (ohne zusätzliche Schichten)
- **Modell 2:** BERT mit eingefrorenen Parametern + CNN-Schichten
- **Modell 3:** BERT mit nicht eingefrorenen Parametern + CNN-Schichten
- **Modell 4:** BERT mit eingefrorenen Parametern + BiLSTM-Schichten
- **Modell 5:** BERT mit nicht eingefrorenen Parametern + BiLSTM-Schichten

Model	Test acc	Train loss	ROC AUC	F1 score
Model 1	0.9579	0.0036	0.9586	0.9607
Model 2	0.9591	0.0200	0.9589	0.9622
Model 3	0.9439	0.0211	0.9449	0.9474
Model 4	0.9614	0.0197	0.9607	0.9646
Model 5	0.9346	0.0227	0.9351	0.9389

Abbildung 2.24: Ergebnisse der verschiedenen Modelle [89]

Wie in Abbildung 2.24 zu sehen, hat das Modell 4 mit einer Test Accuracy von 96,14% und einem F1-Score von 96,46% das beste Ergebnis. Die Kombination aus dem beiden BERT und BiLSTM-Schichten liefert folglich den besten Ansatz zur COVID-19-Fake-News-Erkennung. Die Ergebnisse zeigen, dass sich die semantische Tiefe von BERT und sequentielle Kontextverarbeitung von BiLSTM gut ergänzen.

2.5.7 BERT und LightGBM

[31] entwickelte ein weiteres hybrides Modell welches das BERT-Embedding und ein LightGBM Modell nutzt. Das Modell kombiniert die tiefen semantischen Sprachmerkmale von BERT mit der schnellen, skalierbaren Klassifikationsfähigkeit von LightGBM. Im vorgestellten hybriden Modell wird die Eingabe mittels BERT verarbeitet (siehe Abbildung 2.25). Dabei wird der [CLS]-Token aus den letzten drei Encoderschichten extrahiert und zu einem einzigen Merkmalsvektor zusammengeführt. Dieser Vektor dient als Eingabe für LightGBM, das eine binäre Klassifikation in „wahr“ oder „falsch“ vornimmt.

Genutzt werden in dem Paper folgende drei Datensätze:

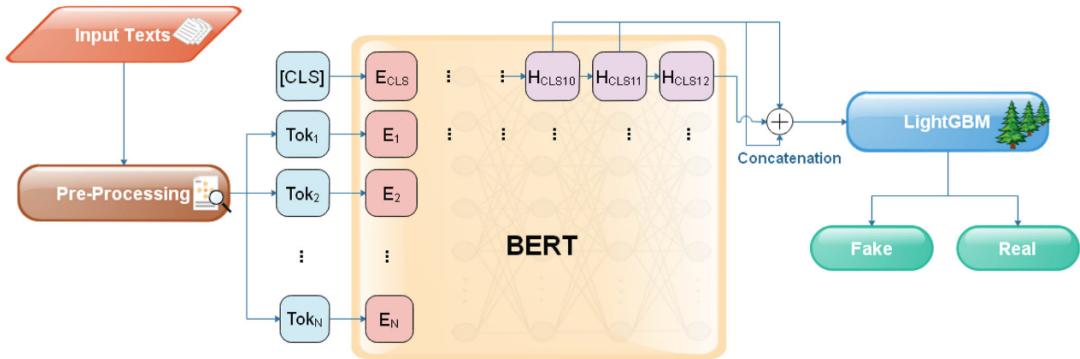


Abbildung 2.25: Architektur des hybriden Modells [31]

- **ISOT:** Enthält ca. 45.000 Artikel, gleichmäßig auf echte und gefälschte Nachrichten verteilt. Die echten Artikel stammen von Reuters, die gefälschten von fragwürdigen Quellen laut PolitiFact. Themenschwerpunkte sind Politik und Weltgeschehen (2016–2017).
- **TI-CNN:** Besteht aus 20.015 Artikeln (8074 echt, 11.941 falsch). Die echten Nachrichten stammen von renommierten Medien wie der New York Times, die gefälschten von über 240 inoffiziellen Webseiten.
- **FNC (Fake News Corpus):** Umfasst Millionen Artikel von über 1000 Domains. Für die Experimente wurde ein ausgeglichener Datensatz aus je 500.000 echten und gefälschten Artikeln erstellt. Enthält zusätzliche Metadaten wie Titel, Autor und URL.

Embed	Model	Title				Text			
		Acc	F1	Pre	Rec	Acc	F1	Pre	Rec
TF-IDF	MNB	82.84	83.21	81.21	85.31	93.39	93.08	97.37	89.15
TF-IDF	LR	82.03	82.17	81.29	83.08	97.01	96.99	97.05	96.94
TF-IDF	LSVM	83.56	83.70	82.71	84.72	97.84	97.83	97.92	97.74
GloVe	MNB	59.88	54.99	62.38	49.16	71.96	71.43	72.58	70.32
GloVe	LSVM	68.07	67.10	68.98	65.32	85.81	85.45	87.36	83.62
GloVe	LSTM	81.63	81.65	81.30	82.00	96.12	96.11	96.07	96.16
BERT	LSTM	86.27	86.29	85.92	86.66	81.69	81.88	80.80	83.00
BERT	Proposed	86.38	86.33	86.36	86.31	99.06	99.05	99.07	99.04

Abbildung 2.26: Ergebnisse der verschiedenen Modelle auf dem FNC-Datensatz [31]

In Abbildung 2.26 zu sehen ist, dass dieses Modell anderen gerade bei großen Datensätzen überlegen ist. [26] nennt dieses Modell im August 2024 mit einer Accuracy von 99,06% die beste State-of-the-Art-Technik.

2.5.8 RoBERTa und LightGBM

Aufbauend auf dem hybriden BERT und LightGBM Model zeigt [84] die Vorteile eines hybriden RoBERTa-LightGBM Modells.

RoBERTa ist eine weiterentwickelte und leistungsstärkere Version von BERT, die durch effizienteres Training und größere Datenbasis bessere Ergebnisse in der Fake-News-Erkennung erzielt. Statt mit den 16GB Textdaten des BERT Modells wurde RoBERTa mit über 160GB trainiert, wodurch eine deutlich bessere Generalisierungsfähigkeit besteht. Außerdem nutzt RoBERTa nicht mehr das statische maskieren (MLM) wie BERT, sondern ein fortgeschrittenes dynamisches maskieren. Hierbei wird jedes mal genau dann ein Maskierungspattern generiert, wenn eine Sequenz einem Modell hinzugefügt wird [55].

Das Paper verwendet den „Fake News Detection Dataset“ zum Training. Die Daten stammen von reuters.com und umfassen jeweils 12.600 Artikel, gesammelt in den Jahren 2016 bis 2017.

Nach der Vorverarbeitung wurden die Daten tokenisiert und mit Roberta in Vektor-Embeddings umgewandelt. Verwendet wird der [CLS] Token aus den letzten drei Hidden Layers, um den gesamten Text zu repräsentieren. Durch Self Attention liefert dies eine feste, dichte Vektor-Repräsentation pro Artikel. Diese Embeddings dienten anschließend als Eingabe für LightGBM zur binären Klassifikation.

RoBERTa-LightGBM erzielt eine Verbesserung gegenüber BERT-LightGBM von bis zu 21% (siehe Abbildung 2.27).

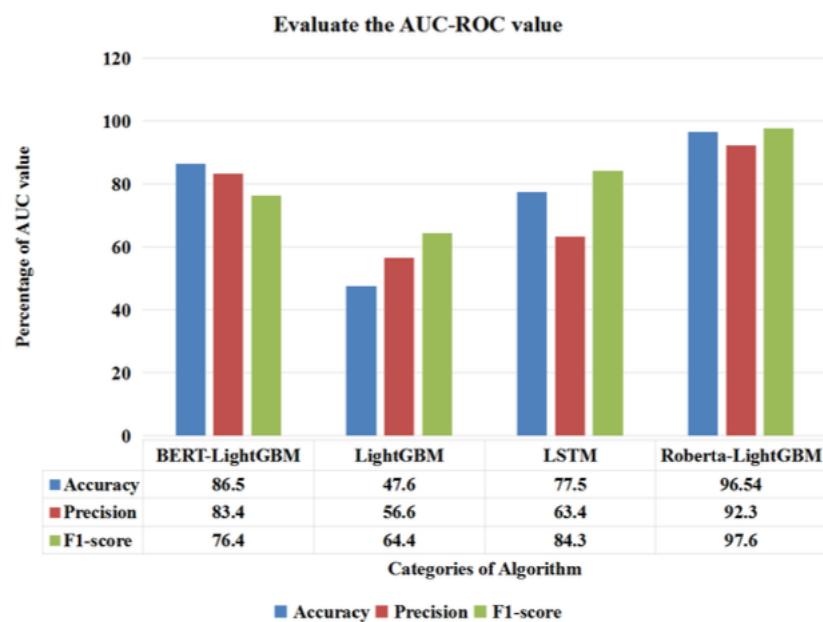


Abbildung 2.27: Vergleich der Ergebnisse der verschiedenen Modelle [84]

3 Relevante Datensätze und Auswahlkriterien

3.1 Vorstellung verfügbarer deutscher Fake-News-Datensätze

In der Tabelle 3.1 sind verschiedene deutsche Fake-News-Datensätze gelistet. Die Eignung bezieht sich darauf, wie sinnvoll die Nutzung dieser jeweiligen Datensätze für das Trainieren eines Modells zum Erkennen von Fake News ist.

3.2 Nutzung englischer Datensätze

Eine in [79] angewandte Lösung ist die Erstellung eines großen Datensatzes mit englischen Artikeln. Das Modell wurde entsprechend auf Englisch trainiert und die deutschen Artikel in der späteren Anwendung vor der Vorhersage übersetzt. Fake News haben oft stilistische, semantische oder rhetorisch manipulierende Muster, welche sich je nach Sprache unterscheiden können. Selbst mit modernen Transformern können diese Muster nicht sicher mit übersetzt werden, da diese einen übersetzen Artikel einer anderen Domäne zuordnen könnten als den ursprünglichen Artikel [41]. Ein typisches Beispiel ist der Satz: „Natürlich wird das RKI bald neue Lockdowns empfehlen – die haben ja auch sonst nichts zu tun.“ Die Übersetzung ins Englische („Of course, the RKI will soon recommend new lockdowns – they have nothing else to do anyway“) wirkt sprachlich korrekt, verliert jedoch die ironisch-sarkastische Eigenschaft. Das Ergebnis ist eine eher sachliche Aussage, welche vom englischen Modell sehr wahrscheinlich nicht mehr als Fake-News klassifiziert wird. Solche stilistischen Abschwächungen sind ein Merkmal von maschinellen Übersetzungen, was sie in diesem Fall problematisch für den Einsatz in der Fake-News-Erkennung macht [52].

Datensatz	Quelle	Anzahl Zeilen	Anteil Fake (%)	Besonderheiten / Einschränkungen	Eignung
Fake News Dataset German	University of Applied Sciences Upper Austria	63.868	7,24 %	Geringer Anteil an Fake-News → unausgewogene Klassenverteilung	Weniger geeignet
German-Fake NC	Fraunhofer-Institut für Sichere Informationstechnologie SIT	489	100 %	Enthält nur Referenzen, keine Texte → keine direkte Textauswertung möglich	Nicht geeignet
FANG-COVID	Association for Computational Linguistics	41.242	31,97 %	Ausgewogene Klassen, vollständige Texte, viele Metadaten (Titel, Header, Label)	Sehr geeignet
DeFaktS	FZI Forschungszentrum Informatik	–	–	Kein Zugang → Nutzung ausgeschlossen	Nicht verfügbar

Tabelle 3.1: Vergleich deutscher Fake-News-Datensätze

3.3 Auswahl und Begründung des finalen Datensatzes

Da der DeFaktS Datensatz nicht öffentlich verfügbar ist fällt dieser aus der Auswahl. Um ein Modell zu trainieren werden viele Daten benötigt. Im Kapitel 2.5.7 umfassen die Datensätze von 20.000 bis hin zu Millionen Artikeln. Deshalb entfällt auch der German-FakeNC Datensatz.

Ein Problem, dass der Fake News Dataset German Datensatz (FNDG) bringt, ist die stark unausgewogene Klassenverteilung. Nur 7,24% der 64.868 Einträge sind als Fake-News gekennzeichnet. Dadurch kann es beim Trainieren eines Modells dazu kommen, dass es überwiegend die häufiger vorkommenden echten Artikel lernt und Fake-News kaum oder gar nicht erkennt. Ein Modell könnte beispielsweise stets „echt“ vorhersagen und damit auf diesem Datensatz eine hohe Accuracy erreichen. Bei späterer Anwendung würde es aber die entscheidenden Fälle nicht mehr differenzieren können. Das führt zu einem guten

Accuracy-Wert, während Recall und F1-Score für die Fake-Klasse deutlich leiden. Das Modell verfehlt somit genau die Einträge, die für die Anwendung zentral sind. Inhaltlich sind die Einträge aber auf einem breiten Gebiet verteilt, was einen Teil dieses Datensatzes für die Auswahl interessant macht.

Der verbleibende FANG-COVID Datensatz [56] fällt aufgrund seiner vielen Einträge und guter Klassenausgewogenheit in die engere Auswahl. Problematisch bei diesem Datensatz ist aber, dass sich alle Einträge im COVID-19 Pandemie Kontext befinden. Es besteht die Gefahr einer thematischen Überanpassung. Das Modell lernt vor allem, typische Begriffe, Erzählmuster und Formulierungen im Zusammenhang mit Pandemie-Fake-News zu erkennen. Dazu können zum Beispiel Impfgegner-Rhetorik, Verschwörungen oder auch Begriffe wie „PCR-Test“, „Lockdown“ oder „RKI“ gehören. In anderen Themenbereichen wie Politik, Migration oder Klima erkennt es dagegen manipulierte Inhalte unter Umständen nicht, da es diese Muster nie gelernt hat. Außerdem kann das Modell semantische Verzerrungen entwickeln. Wörter, die in Pandemie-Fake-News häufig auftreten, könnten fälschlich als Indiz für eine Fake Klassierung gewertet werden, auch wenn sie in neutralem oder echtem Kontext auftreten [14, 59]. Dadurch steigt die Gefahr von False Positives bei echten Nachrichten außerhalb des COVID-Kontexts. Ein Modell, das nur mit COVID-bezogenen Daten trainiert wurde, kann inhaltlich und stilistisch stark eingeschränkt generalisieren und verfehlt damit das Ziel, Fake News themenübergreifend zuverlässig zu erkennen.

Durch die Analyse der Datensätze ergeben sich folgende Möglichkeiten für das Trainieren eines Modells zur Fake-News Erkennung:

- Nur FANG-COVID mit allen 41.242 Einträgen, aber 31,97% Klassenausgewogenheit
- FANG-COVID mit ca. 26.000 Einträgen aber dafür 50/50 Klassenausgewogenheit
- FANG-COVID und FNDG kombiniert für bessere Generalisierung (105.110 Einträge, aber 16,94% Klassenausgewogenheit)
- FANG-COVID und alle Fake Artikel von FNDG kombiniert für bessere Klassenausgewogenheit und mehr Daten (45.866 Einträge mit 38,82% Klassenausgewogenheit)
- Alle Fake Einträge aus FANG-COVID und FNDG kombiniert (17.809 Fake Einträge) kombiniert mit Stichproben aus sowohl FANG-COVID als auch FNDG um

3 Relevante Datensätze und Auswahlkriterien

eine sinnvolle Klassenausgewogenheit mit vielen Daten und einer guten Generalisierung zu erreichen (45.866 Einträge mit 38.82% Klassenausgewogenheit, aber 28.057 echten Einträgen zu 50/50 aus FANG-COVID und FNDG).

4 Konzeption der Softwarelösung

4.1 Machine Learning Modell

4.1.1 Auswahl und Begründung der genutzten Modelle

Um ein Modell zu entwickeln, welches möglichst gute Metriken erzielt, müssen verschiedenen Modelkombinationen getestet werden. Basierend auf den Arbeiten von [31] und [84], welche Accuracy Werte bis zu 99,06% erreichten, werden in dieser Arbeit weitere BERT und RoBERTa Modelle mit LightGBM kombiniert.

Der wesentliche Vorteil beim Kombinieren der Transformer Modelle mit dem LightGBM Modell liegt in den verschiedenen Stärken der beiden Ansätze:

Transformer Modelle wie BERT, bzw. RoBERTa extrahieren Sprachrepräsentationen und erfassen dabei den Kontext eines Wortes im Satz, indem sie sowohl den vorhergehenden als auch den nachfolgenden Text berücksichtigen. Dabei wird der vollständige sprachliche Zusammenhang eines Tokens innerhalb des gesamten Satzes oder Dokuments erkannt.

Im Gegensatz dazu ist LightGBM ein leistungsstarker, baumbasierter Klassifikator. Die Stärken dieses Modells liegen in Effizienz, Skalierbarkeit und Robustheit. Es arbeitet besonders gut bei tabellarischen, hochdimensionalen Feature-Repräsentationen. Diese können zum Beispiel durch BERT-Embeddings entstehen. Außerdem erfolgt die finale Klassifikation über LightGBM mit deutlich weniger Rechenaufwand, da keine weitere tiefere neuronale Architektur benötigt wird.

Beide Arbeiten zeigen, dass die Kombination zu besserer Generalisierung, niedrigerem Overfitting und schnellerem Training führt.

Folgende Kombinationen werden im Folgenden verglichen:

- BERT und LightGBM

- RoBERTa und LightGBM
- XML-RoBERTa und LightGBM

4.1.2 Datenvorverarbeitung

Der FNDG Datensatz enthält die Merkmale 'id', 'url', 'Titel', 'Body', 'Kategorie', 'Datum', 'Quelle', 'Fake' und 'Art'. Hiervon beinhalten die drei Merkmale 'Titel', 'Body' und 'Fake' die relevanten Informationen für diese Anwendung. In 'Titel' steht die jeweilige Überschrift des Artikels und in 'Body' der eigentliche Inhalt. Das Merkmal 'Fake' gibt in der Nominalskala an ob der Artikel als gefälscht klassifiziert ist oder nicht (1 für Fake, 0 für echt).

Im FANG-COVID Datensatz sind die Merkmale 'Unnamed: 0.1', 'Unnamed: 0', 'article', 'date', 'header', 'label', 'url', 'hist', 'tweeet', 'repl', 'retw', 'like' und 'quote' enthalten. 'article', 'header' und 'label' sind hierbei relevant. 'header' und 'article' sind analog zum FNDG Datensatz Überschrift und Inhalt des Artikels. 'label' enthält entweder den Wert 'real' um den Artikel als echt oder 'fake' um den Artikel als gefälscht zu klassifizieren.

Die beiden Datensätze werden zur Weiterverarbeitung zusammengefasst. Überschrift und Inhalt werden konkateniert, da das RoBERTa Modell nur einen Wert pro Label verarbeiten kann. Dieses Merkmal wird umbenannt zu 'text'. Für die Klassifizierungskennzeichnung wird die Nominalskala des FNDG Datensatzes und der Titel des FANG-Covid Datensatzes übernommen.

Der finale Datensatz umfasst 45869 Artikel mit jeweils dessen *text* und *label*. 38,83% dieser Artikel (17.813) sind Fake.

Die Aufteilung der Daten erfolgt in Trainings-, Validierungs- und Testdatensätze. 80% des Gesamtdatensatzes werden für das Training genutzt und jeweils 10% für das Validierung und Testen.

4.1.3 Fine-tuning der Transfomer Modelle

Transformer Modelle wie zum Beispiel BERT sind in der Regel vortrainiert. Dabei werden sie auf großen Mengen unannotierter Textdaten trainiert, um ein allgemeines Sprachverständnis zu erlernen. Dieser gesamte Prozess erfolgt unüberwacht. Um die vortrainierten Transformer Modelle für die Fake-News Erkennung nutzbar zu machen, müssen sie

einmalig an die spezifische Aufgabe angepasst werden (*Fine-Tuning*). Hierfür wird das Modell auf dem in Kapitel 4.1.2 erzeugtem Datensatz weitertrainiert. Dabei wird die zugrundeliegenden Sprachrepräsentationen auf die konkrete Zielaufgabe, der Fake-News Klassifizierung übertragen.

4.1.4 Erzeugung der Embeddings

Nach der Aufteilung des Datensatzes, werden diese Daten von den Transformer Modellen in Embeddings umgewandelt. Zuerst werden die Eingaben dafür in Token zerlegt. BERT nutzt den WordPiece Tokenizer, RoBERTa das Byte-Pair-Encoding und XML-RoBERTa den SentencePiece Tokenizer. Alle diese Tokenizer zerteilen Wörter unterschiedlich, was zu verschiedenen Tokenfolgen und damit auch zu unterschiedlichen Embedding-Repräsentationen führt.

Eine wichtige Eigenschaft der drei Transformer Modelle ist, dass nach dem Tokenisieren maximal 512 Token pro Eingabe verarbeitet werden können. Was genau ein Token ist, hängt in diesem Fall vom genutzten Tokenizer ab. Es kann sowohl ein Wort als auch nur ein Wortfragment sein. Alle Token einer zu verarbeitenden Eingabe, die nach dem 512. Token kommen, werden abgeschnitten. Eingabesequenzen, welche kürzer als 512 Token sind, werden mit angehängten [PAD]-Tokens aufgefüllt, damit alle Eingaben die gleiche Länge haben.

[81] testet verschiedene *Truncation*-Methoden und zeigt für BERT, dass bei Filmrezensionen und Nachrichtenartikeln eine Zusammensetzung der ersten 256 und letzten 256 Token am effektivsten ist. Der Unterschied zur Nutzung der ersten 512 Token ist aber minimal, daher wird in dieser Arbeit letzteres implementiert.

Die erzeugten Token werden von den jeweiligen angepassten Transformer Modellen in dichte Vektoren verarbeitet. Diese Vektoren bilden die Grundlage für die nachfolgenden *Self-Attention*-Mechanismen, die kontextabhängige, semantisch reichhaltige Embeddings erzeugen. Diese Embeddings entstehen über mehrere aufeinanderfolgende *hidden layers*, wobei jede Schicht die Werte weiter verfeinert und anreichert.

Um die Ausgabe der verschiedenen Embeddings der *hidden layer* zusammenzufassen kann zum Beispiel der [CLS]-Token aus einer oder mehreren Schichten extrahiert oder ein Durchschnitt aller Token-Embeddings gebildet werden.

[81] zeigt für BERT, dass die letzten Layer die meisten Informationen beinhalten und besonders die Kombination der letzten vier Layer mit anschließender Bildung des Maximums das beste Ergebnis erzielt. In dieser Arbeit werden verschiedene Kombinationen getestet.

4.1.5 Nutzung der Embeddings im LightGBM Modell

Der für das Trainieren des LightGBM Modells benötigte Datensatz setzt sich aus der jeweiligen Zusammenfassung der *hidden layer* und dem Klassifizierungslabel des Artikels, durch welches die Embeddings erzeugt wurden, zusammen.

Für das Training werden die Artikel des in Kapitel 4.1.2 erzeugten Datensatzes gemappt. Jeder Artikel wird hierbei durch ein, vom angepassten Transformer Modell erzeugten zusammengefassten Embedding ersetzt. Das entsprechende Label wird als zweite Spalte ergänzt.

Das trainierte LightGBM Modell kann anschließend neu erzeugte Embeddings effizient klassifizieren.

4.1.6 Gesamtarchitektur

Für die finale Anwendung dieser Arbeit wird ein Webserver aufgesetzt, welcher Nachrichtenartikel empfängt und diese anhand der konfigurierten Modelle tokenisiert, eingebettet und klassifiziert (siehe Abbildung 4.1).

4.2 Webagent

Der Webagent hat die Aufgabe die Artikel auf den verschiedenen Nachrichtenportalen zu lesen und zu ergänzen. Hierfür muss erkannt werden auf welcher Seite sich der Nutzer befindet. Außerdem muss das html dieser Seite ausgelesen und analysiert werden können.

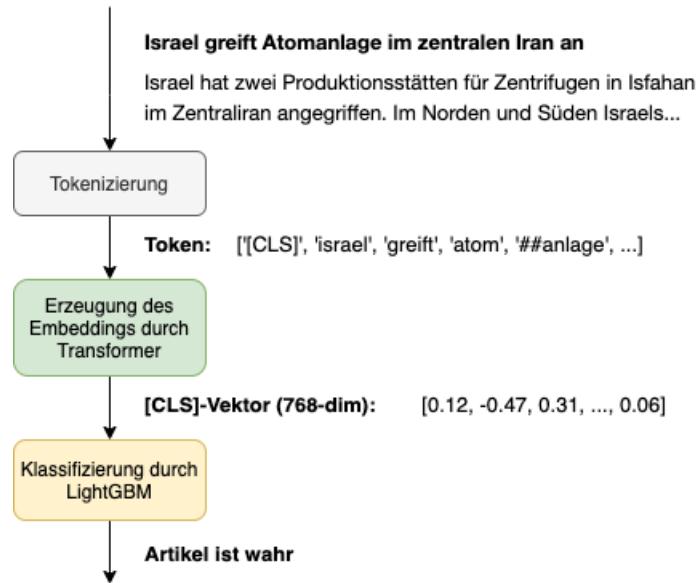


Abbildung 4.1: Beispielhafter Ablauf einer Klassifizierung eines Artikels

Als Beispiel die Seite Bild.de: Je nach Fenstergr e e hat die Seite entweder die Dom ne <https://www.bild.de/> oder <https://m.bild.de/>.

Die Startseite ist wie folgt aufgebaut:

```

<!DOCTYPE html>
<html>
  <head>
    ...
  </head>
  <body>
    <div id="app">
      ...
      <div id="page-content">
        <header/>
        <main>
          <!-- Es gibt auf der Startseite
          ueber 50 dieser section-Elemente -->
          <section>
            <article />
          </section>
        </main>
      </div>
    </body>
  </html>

```

```

        </section>
        </main>
        <footer />
    </div>
</div>
</body>
</html>

```

Wenn ein Artikel geöffnet ist, ist der DOM dem der Startseite sehr ähnlich. Der einzige wesentliche Unterschied ist, dass im *main*-Element nur noch ein *article*-Element ist und nicht beliebig viele *section*-Elemente. Ob ein Artikel geöffnet ist, kann also anhand der Anzahl der *article*-Elemente bestimmt werden.

```

<article>
    <h2 class="document-title-document-title-article">
        <span class="kicker">Kicker des Artikels</span>
        <span class="headline">Titel des Artikels</span>
    </h2>
</article>
<div class="article-body">
    <!-- Pro Artikel gibt es ca. 10 p-Elemente -->
    <p>Inhalt des Artikels</p>
</div>

```

Der Titel und Inhalt des Artikels kann den entsprechenden html-Elementen entnommen werden. Diese werden anschließend an die API gesendet und dort verarbeitet. Der Rückgabewert der API enthält dann die Info ob der Artikel falsch oder echt ist. Diese wird in einem vom Webagent erzeugten *div*-Container über dem Artikel eingefügt.

Zur Bestimmung des geeigneten Tools für diese Anforderungen, wurden verschiedene Technologien verglichen (siehe Tabelle A.2). Aufgrund des begrenzten Zugriffs auf die zu analysierende Seiten, bieten sich die beiden Client-seitigen Umsetzungen eine Chrome Extension zu implementieren oder über Tampermonkey Userscripts auszuführen am ehesten an.

Im Vergleich zu Userscripts unterstützt die Extension mehrere Komponenten (Content Scripts, Background Scripts, Popup, Optionsseite). Anhand dieser können der DOM beobachtet, ein persistenter Speicher genutzt, Kontextmenüs erstellt und auf Browserak-

tionen reagiert werden (z.B. Tabwechsel, Navigation). Ein Userscript hingegen ist ein einfaches Script, das nur beim Laden einer Seite aktiv ist und dementsprechend keine Hintergrundverarbeitung und keine erweiterten UI-Komponenten zur Verfügung stellt.

Zur Implementierung des Webagents wird also eine Chrome Extension genutzt.

5 Umsetzung des Prototyps

5.1 Implementierung des Machine Learning Modells

5.1.1 Fine-tuning der Transformer Modelle

Für das *fine-tuning* (Training) der verschiedenen Modelle werden die Transformers-Bibliothek von Hugging Face, Python und das Deep-Learning-Framework PyTorch verwendet.

Trainiert werden fünf verschiedene Transformer Modelle mit dem Datensatz aus Kapitel 4.1.2:

- Bert base
- RoBERTa base
- RoBERTa large
- XLM-RoBERTa base
- XLM-RoBERTa large

In A.3 angehängt, findet sich ein tabellarischer Vergleich der verschiedenen BERT- und RoBERTa-Modelle mit deren verschiedenen technischen Eigenschaften.

Das Tokenisieren erfolgt über den jeweiligen mitgelieferten Tokenizer.

Eine Übersicht der für das *fine-tuning* genutzten Hyperparameter findet sich im Anhang (siehe Tabelle A.4).

Zur dynamischen Anpassung der Lernrate während des Trainings, wird ein linearer Learning Rate Scheduler mit Warmup verwendet. Zu Beginn des Trainings wird die Lernrate

über 10% der Trainingsschritte schrittweise erhöht, um das Modell zu stabilisieren. Anschließend wird sie linear bis zum Ende des Trainings wieder abgesenkt. Dieses Vorgehen hilft dabei, Konvergenzprobleme zu vermeiden und die Modellleistung zu verbessern. Konvergenzprobleme entstehen, wenn das Modell beim Training nicht richtig lernt und die Fehler (*loss*) nicht zuverlässig kleiner werden.

Das Training wird auf einer Google Colab Laufzeit mit einer A100 GPU mit erweitertem RAM Speicher durchgeführt. Die trainierten Modelle werden jeweils als einzelnen Repository im *Hugging Face Hub* gespeichert.

5.1.2 Erzeugung der Embeddings

Für das Erzeugen der Embeddings wird der Datensatz aus Kapitel 4.1.2 in Trainings- (80%) und Testdaten (20%) aufgeteilt und tokenisiert.

Diese Datensätze durchlaufen anschließend folgende Schritte:

1. **Vorbereitung des Modells:** Das Transformer Modell wird in den Evaluationsmodus gesetzt und auf die A100 GPU verschoben, um eine effiziente Berechnung sicherzustellen.
2. **Erstellung eines *DataLoaders*:** Die tokenisierten Eingabesequenzen (bestehend aus `input_ids`, `attention_mask` und `labels`) werden zu Batches zusammengefügt, welche von der GPU parallel verarbeitet werden können. Diese werden in einem sogenannten, von *PyTorch* zur Verfügung gestelltem, *DataLoader* gespeichert.
3. **Vorwärtspass:** Die Batches des *DataLoaders* werden in das Modell eingespeist, um Vorhersagen bzw. Zwischenrepräsentationen zu berechnen. Hierbei wird keine Gradientenberechnung (Backpropagation) durchgeführt, das Modell lernt also nicht von den Eingaben.
4. **Extraktion der *hidden states*:** Für jedes Batch wird die Ausgabe aller versteckten Schichten (`hidden layer`) der Transformer Modelle berechnet. Jedes `hidden layer` erzeugt einen `hidden state`.
5. **Pooling über die letzten vier Schichten:** Die letzten vier `hidden state`-Schichten werden ausgewählt und über eine Mittelwertbildung (Average Pooling) zusammengefasst, um eine robustere Token-Repräsentation zu erhalten.

6. **Maskierung und Mittelung über Tokens:** Mithilfe der `attention_mask` werden `[PAD]`-Tokens ausgeschlossen. Die verbleibenden Token-Embeddings werden über die Sequenzlänge gemittelt, sodass ein einziger Vektor pro Eingabesequenz entsteht.
7. **Speicherung der Embeddings:** Die resultierenden Satz-Embeddings sowie die zugehörigen Labels werden gesammelt und in *Numpy*-Arrays konvertiert.
8. **Finaler Output:** Nach Verarbeitung aller Batches werden die Satz-Embeddings und Labels aller Beispiele zu zwei großen Arrays (Embeddings und Label) zusammengefügt.

Nach Durchlaufen der Schritte werden die Embeddings und Label der Trainings- und Testdaten in einem Python *Dictionary* gespeichert.

5.1.3 Nutzung der Embeddings im LightGBM Modell

Die Python LightGBM-Bibliothek stellt ein einfach zu implementierendes Modell zur Verfügung. Dieses wird auf den verschiedenen erzeugten Embeddings trainiert.

Eine Übersicht der im Modell genutzten Hyperparameter findet sich im Anhang (siehe Tabelle A.5). Zur Wahl der geeigneten Hyperparameter wird einmalig die *Optuna*-Bibliothek genutzt. Dabei werden mithilfe von K-Fold-Cross-Validation (siehe Abbildung 5.1) verschiedene Kombinationen von LightGBM-Hyperparametern getestet, um das Modell mit dem höchsten F1-Score zu finden.

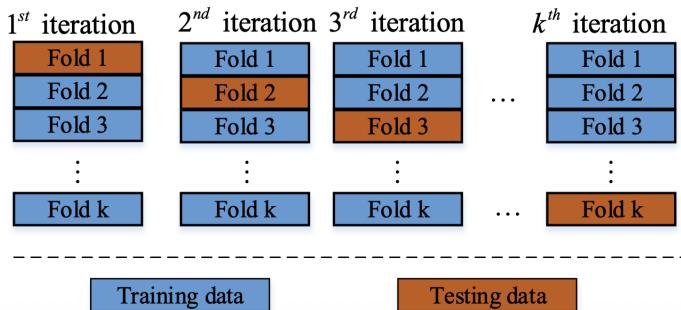


Abbildung 5.1: K-Fold-Cross-Validation [61]

Anhand von dem, in der Bibliothek zur Verfügung gestellten, *early-stopping* wird das Training des Modells überwacht und frühzeitig gestoppt, wenn bestimmte Anforderungen

erreicht sind. Die Parameter sind in diesem Fall so definiert, dass das Training beendet wird, sobald sich der logarithmische Verlust (*Logloss*) auf den Validierungsdaten über 100 aufeinanderfolgende Iterationen hinweg nicht mehr verbessert.

5.1.4 Gesamtarchitektur

5.2 Implementierung der Chrome Erweiterung

Die Chrome Erweiterung wurde mit der Version Manifest 3 implementiert. Genutzt wurden ein *Service Worker* ein *Content Script* pro Nachrichtenportal und ein *Popup*.

Service Worker steuern eine Seite genau dann, wenn ein Service Worker auf dieser Netzwerkanfragen in seinem Namen abfangen kann. Der Service Worker kann dann Aufgaben für die Seite innerhalb eines bestimmten Scopes ausführen

Der Lifecycle eines Service Workers ist in folgende Events unterteilt: installing, installed, activating, activated.

Nach Abschluss der Aktivierung steuert der Service Worker die Seite standardmäßig erst bei der nächsten Navigation oder Seitenaktualisierung [35].

Content Scripts sind Dateien, die im Kontext von Webseiten ausgeführt werden. Mit dem standardmäßigen Document Object Model (DOM) können sie Details der Webseiten lesen, die der Browser besucht, Änderungen daran vornehmen und Informationen an die übergeordnete Erweiterung weitergeben [33].

Die Kommunikation mit den Service Worker erfolgt über die Extension-API *runtime*.

Pop-ups sind Aktionen, bei denen ein Fenster angezeigt wird, über das Nutzer mehrere Erweiterungsfunktionen aufrufen kann. Sie werden durch ein Tastenkürzel, durch Klicken auf das Aktionssymbol der Erweiterung oder durch Drücken von chrome.action.openPopup() ausgelöst. Pop-ups werden automatisch geschlossen, wenn der Nutzer sich auf einen Bereich des Browsers außerhalb des Pop-ups konzentriert [34].

5 Umsetzung des Prototyps

In Abbildung A.3 zu sehen ist das Sequenzdiagramm des Webagents. Wie in Kapitel 4.2 beschrieben wird zuerst die URL geprüft. Erfüllt diese die vorgegebenen Bedingungen wird der geöffnete Artikel gelesen und von einer weiteren Anwendung analysiert. Anschließend wird das Ergebnis der Analyse in einem *div*-Container über dem Artikel eingefügt.

Um die Veränderungen im Browser zu überwachen wurde die *tabs*-API von Chrome genutzt. Anhand dieser kann das Tab-System eines Browsers überwacht und zum Beispiel auch auf jede Veränderung der URL reagiert werden. Außerdem ermöglicht die API das Versenden von Nachrichten an alle aktiven Content Scripts. Diese werden dann im jeweiligen Content Script über die *runtime*-API empfangen und ausgelesen.

6 Evaluation und Ergebnisse

Modell	Accuracy	F1-Score	Loss	Eval-Zeit (s)
XLM-RoBERTa-Large	0.9795	0.9795	0.1070	13.96
RoBERTa-Large	0.9765	0.9764	0.1395	14.04
RoBERTa-Base	0.9751	0.9751	0.1273	5.86
XLM-RoBERTa-Base	0.9717	0.9716	0.1527	5.77
BERT-Base-Uncased	0.9533	0.9531	0.1952	6.33

Tabelle 6.1: Direktes Fine-Tuning – Testergebnisse der Modelle

Embedding-Modell	Accuracy	F1-Score
XLM-RoBERTa-Large	0.9784	0.9773
XLM-RoBERTa-Base	0.9753	0.9738
RoBERTa-Large	0.9753	0.9738
RoBERTa-Base	0.9729	0.9713
BERT-Base-Uncased	0.9472	0.9439

Tabelle 6.2: LightGBM-Ergebnisse auf Embeddings der Modelle

Modell	Transformer		LightGBM	
	Accuracy	F1	Accuracy	F1
XLM-RoBERTa-Large	0.9795	0.9795	0.9784	0.9773
RoBERTa-Large	0.9765	0.9764	0.9753	0.9738
RoBERTa-Base	0.9751	0.9751	0.9729	0.9713
XLM-RoBERTa-Base	0.9717	0.9716	0.9753	0.9738
BERT-Base-Uncased	0.9533	0.9531	0.9472	0.9439

Tabelle 6.3: Accuracy und F1-Score: Transformer Fine-Tuning vs. LightGBM auf Embeddings

7 Fazit

8 Ausblick

- Verwendung eines aktuelleren Datensatzes mit domänenübergreifenden Themen - Verwendung eines RoBERTa Modells, welches auch mit deutschen Daten vorgenutzt wurde.
- Verwendung des RoBERTA Modells "xml-roberta-xl Nutzung von Modellen mit erhöhter Inputlänge (mehr als 512 Token) oder Kombination verschiedener Eingaben (head + tail [81]) - Vergleich mehrerer Modelle - Implementierung FakeBert mit XML-RoBERTa
- Deployment der Anwendung und Integration mehrerer Nachrichtenportalen - Entwicklung einer eigenen Website in welcher die Nachrichtenartikel gepostet werden können.

Literaturverzeichnis

- [1] Alan Agresti. *An Introduction to Categorical Data Analysis*. Wiley, 3rd edition, 2018. ISBN 9781119405269.
- [2] AIML. Compare the different sequence models: Rnn, lstm, gru, and transformers, April 2025. URL <https://aiml.com/compare-the-different-sequence-models-rnn-lstm-gru-and-transformers/>. Zuletzt abgerufen am 20. Mai 2025.
- [3] Mutaz Al-Tarawneh, Ashraf Al-Khresheh, Omar Al-Irr, Ajla Kulaglic, Kassem Danach, Hassan Kanj, and Ghayth Almahadin. Towards accurate fake news detection: Evaluating machine learning approaches and feature selection strategies. *European Journal of Pure and Applied Mathematics*, 18, 05 2025. doi: 10.29020/nybg.ejpam.v18i2.6087.
- [4] F. Alzami, E. D. Udayanti, and D. P. Prabowo. Document preprocessing with tf-idf to improve the polarity classification performance. *Kinetik Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, 2020. URL <https://scholar.archive.org/work/pfc1h2k6tffsdeh5ezp3qtdkoy/access/wayback/http://202.52.52.28/index.php/kinetik/article/download/1066/pdf>.
- [5] Ashish, Sonia, Monika Arora, Hemraj, Anurag Rana, and Gaurav Gupta. An analysis and identification of fake news using machine learning techniques. In *2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 634–638, 2024. doi: 10.23919/INDIACom61295.2024.10498879.
- [6] Naila Aslam, Kewen Xia, Furqan Rustam, Afifa Hameed, and Imran Ashraf. Using aspect-level sentiments for calling app recommendation with hybrid deep-learning models. *Applied Sciences*, 12:8522, 08 2022. doi: 10.3390/app12178522.

- [7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016. URL <https://arxiv.org/abs/1409.0473>.
- [8] Joao Pedro Baptista and Anabela Gradim. Understanding fake news consumption: A review. *Social Sciences*, 9(10), 2020. ISSN 2076-0760. doi: 10.3390/socsci9100185. URL <https://www.mdpi.com/2076-0760/9/10/185>.
- [9] Joel Barnard. Was sind worteinbettungen?, 2024. <https://www.ibm.com/de-de/think/topics/word-embeddings> [Accessed: 18.05.2025].
- [10] A. Berrajaa. Natural language processing for the analysis sentiment using a lstm model. *International Journal of Advanced Computer Science and Applications*, 13 (5), 2022. doi: 10.14569/IJACSA.2022.0130589. URL <https://doi.org/10.14569/IJACSA.2022.0130589>.
- [11] Hendrik Blockeel, Laurens Devos, Benoît Frénay, Géraldin Nanfack, and Siegfried Nijssen. Decision trees: from efficient prediction to responsible ai. *Frontiers in Artificial Intelligence*, Volume 6 - 2023, 2023. ISSN 2624-8212. doi: 10.3389/frai.2023.1124553. URL <https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2023.1124553>.
- [12] Manan Buddhadev and Virtee Parekh. Fake news detection: Benchmarking machine learning and deep learning approaches. *ESP Journal of Engineering and Technology Advancements*, 5:39–46, 04 2025. doi: 10.56472/25832646/JETA-V5I2P106.
- [13] Michael Bürker. Fake-news, propaganda & co: Wie behalte ich den überblick?, 2022. URL <https://www.haw-landshut.de/aktuelles/beitrag/fake-news-propaganda-co-wie-behalte-ich-den-ueberblick>. Interview geführt von EINFALLSreich, Hochschule Landshut.
- [14] Ben Chen, Bin Chen, Dehong Gao, Qijin Chen, Chengfu Huo, Xiaonan Meng, Weijun Ren, and Yang Zhou. Transformer-based language model fine-tuning methods for covid-19 fake news detection, 2023. URL <https://arxiv.org/abs/2101.05509>.
- [15] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

- tion for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939785. URL <https://doi.org/10.1145/2939672.2939785>.
- [16] KENNETH WARD CHURCH. Word2vec. *Natural Language Engineering*, 23(1): 155–162, 2017. doi: 10.1017/S1351324916000334.
- [17] Piotr Cichosz. A case study in text mining of discussion forum posts: Classification with bag of words and global vectors. *International Journal of Applied Mathematics and Computer Science*, 2018. URL <https://sciendo.com/pdf/10.2478/amcs-2018-0060>.
- [18] Alexis CONNEAU and Guillaume Lample. Cross-lingual language model pretraining. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/c04c19c2c2474dbf5f7ac4372c5b9af1-Paper.pdf.
- [19] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116, 2019. URL <http://arxiv.org/abs/1911.02116>.
- [20] IBM Corporation. Bag of words explained. <https://www.ibm.com/think/topics/bag-of-words>, 2024. Accessed: 2025-05-16.
- [21] M. Das and P. J. A. Alphonse. A comparative study on tf-idf feature weighting method and its analysis using unstructured dataset. *arXiv preprint arXiv:2308.04037*, 2023. URL <https://arxiv.org/pdf/2308.04037.pdf>.
- [22] DataCamp. What is a confusion matrix in machine learning. <https://www.datacamp.com/de/tutorial/what-is-a-confusion-matrix-in-machine-learning>, 2025. Abgerufen am 1. Juni 2025.
- [23] N. Deshai and B. Bhaskara Rao. Unmasking deception: a cnn and adaptive pso approach to detecting fake online reviews. *Soft Computing*, 27(16):11357–11378, 2023. doi: 10.1007/s00500-023-08507-z. URL <https://doi.org/10.1007/s00500-023-08507-z>.

- [24] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL <http://arxiv.org/abs/1810.04805>.
- [25] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL <https://arxiv.org/abs/1810.04805>.
- [26] Pummy Dhiman, Amandeep Kaur, Deepali Gupta, Sapna Juneja, Ali Nauman, and Ghulam Muhammad. Gbert a hybrid deep learning model based on gpt-bert for fake news detection. *Helicon*, 10(16), 2024. doi: 10.1016/j.heliyon.2024.e35865. URL <https://doi.org/10.1016/j.heliyon.2024.e35865>.
- [27] Giacomo Domeniconi, Gianluca Moro, Roberto Pasolini, and Claudio Sartori. A comparison of term weighting schemes for text classification and sentiment analysis with a supervised variant of tf.idf. In *A Comparison of Term Weighting Schemes for Text Classification and Sentiment Analysis with a Supervised Variant of tf.idf*, volume 584, pages 39–, 02 2016. ISBN 978-3-319-30162-4. doi: 10.1007/978-3-319-30162-4_4.
- [28] SKOPOS ELEMENTS. Von woertern zur bedeutung: Wie word embeddings die sprachverarbeitung revolutionieren, 2023. <https://skopos-elements.de/wissen/blog/maschinelles-lernen/word-embeddings> [Accessed: 18.05.2025].
- [29] Abdelilah Elhachimi, Eddabbah Mohamed, Abdelhafid Benksim, Hamid Ibanni, and Mohamed Cherkaoui. Machine learning-based prediction of cannabis addiction using cognitive performance and sleep quality evaluations. *International Journal of Advanced Computer Science and Applications*, 16, 04 2025. doi: 10.14569/IJACSA.2025.0160439.
- [30] B. B. Elov, S. M. Khamroeva, and R. H. Alayev. Methods of processing the uzbek language corpus texts. *Journal of Open Innovations*, 2023. URL <https://cyberleninka.ru/article/n/methods-of-processing-the-uzbek-language-corpus-texts>.
- [31] Ehab Essa, Karima Omar, and Ali Alqahtani. Fake news detection based on a hybrid bert and lightgbm models. *Complex & Intelligent Systems*, 9(6):6581–6592, 2023. doi: 10.1007/s40747-023-01098-0. URL <https://doi.org/10.1007/s40747-023-01098-0>.

- [32] Hugging Face. Wordpiece tokenization, 2025. <https://huggingface.co/learn/llm-course/chapter6/6> [Accessed: 18.05.2025].
- [33] Chrome for Developers. Content scripts. <https://developer.chrome.com/docs/extensions/develop/concepts/content-scripts>, 2025. Zugriff am 11. Mai 2025.
- [34] Chrome for Developers. Add popup. <https://developer.chrome.com/docs/extensions/develop/ui/add-popup>, 2025. Zugriff am 11. Mai 2025.
- [35] Chrome for Developers. A service worker's life. <https://developer.chrome.com/docs/workbox/service-worker-lifecycle>, 2025. Zugriff am 11. Mai 2025.
- [36] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2nd edition, 2019. ISBN 978-1492032649.
- [37] Benyamin Ghogogh and Ali Ghodsi. Attention mechanism, transformers, bert, and gpt: Tutorial and survey. working paper or preprint, December 2020. URL <https://hal.science/hal-04637647>.
- [38] Katrin Hartwig and Christian Reuter. *Fake News technisch begegnen – Detektions- und Behandlungsansätze zur Unterstützung von NutzerInnen*, pages 133–149. Springer Fachmedien Wiesbaden, Wiesbaden, 2021. ISBN 978-3-658-32957-0. doi: 10.1007/978-3-658-32957-0_7. URL https://doi.org/10.1007/978-3-658-32957-0_7.
- [39] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2nd edition, 2009. ISBN 9780387848570.
- [40] Jakob Henke. *Nachrichten im Auge des Betrachters: Der Selektionsprozess aus Perspektive der Nutzer*innen*. Springer VS, Wiesbaden, Deutschland, 2024. ISBN 978-3-658-46607-7. doi: 10.1007/978-3-658-46608-4. URL <https://doi.org/10.1007/978-3-658-46608-4>.
- [41] Jiwoo Hong, Yejin Cho, Jaemin Jung, Jiyoung Han, and James Thorne. Disentangling structure and style: Political bias detection in news by inducing document hierarchy, 2023. URL <https://arxiv.org/abs/2304.02247>.

- [42] Benjamin Horne and Sibel Adali. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. *Proceedings of the International AAAI Conference on Web and Social Media*, 11, 03 2017. doi: 10.1609/icwsm.v11i1.14976.
- [43] Junfeng Hu, Xiaosa Li, Yuru Xu, Shaowu Wu, and Bin Zheng. Evaluation of company investment value based on machine learning, 2020. URL <https://arxiv.org/abs/2010.01996>.
- [44] Vivek Iyer. A comparative analysis of sentiment classification models for improved performance optimization. *NHSJS (National High School Journal of Science)*, 2024. URL <https://nhsjs.com/wp-content/uploads/2024/05/A-Comparative-Analysis-of-Sentiment-Classification-Models-for-Improved-Performance-Optimization.pdf>.
- [45] Vikramaditya Jakkula. Tutorial on support vector machine (svm). *School of EECS, Washington State University*, 37(2.5):3, 2006.
- [46] Daniel Jurafsky and James H. Martin. *Chapter 5: Logistic Regression*, chapter 5, page n/a. Stanford University, 2025. URL <https://web.stanford.edu/~jurafsky/slp3/>. Online draft.
- [47] Rohit Kumar Kaliyar, Anurag Goswami, and Pratik Narang. Fakebert: Fake news detection in social media with a bert-based deep learning approach. *Multimedia Tools and Applications*, 80(8):11765–11788, 2021. doi: 10.1007/s11042-020-10183-2. URL <https://doi.org/10.1007/s11042-020-10183-2>.
- [48] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm a highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.
- [49] KiKaBeN. Transformers encoder decoder, December 2021. URL <https://kikaben.com/transformers-encoder-decoder/>. Veröffentlicht am 12. Dezember 2021.

- [50] Juhani Kivimäki, Jakub Bialek, Wojtek Kuberski, and Jukka K. Nurminen. Performance estimation in binary classification using calibrated confidence, 2025. URL <https://arxiv.org/abs/2505.05295>.
- [51] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Eduardo Blanco and Wei Lu, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL <https://aclanthology.org/D18-2012/>.
- [52] Ramona Kühn, Jelena Mitrović, and Michael Granitzer. Enhancing rhetorical figure annotation: An ontology-based web application with rag integration, 2024. URL <https://arxiv.org/abs/2412.13799>.
- [53] Wolfgang Lieb. *Wandel des Mediensystems – Kann das Internet die klassischen Medien ergänzen oder gar ersetzen?*, pages 291–319. Springer Fachmedien Wiesbaden, Wiesbaden, 2023. ISBN 978-3-658-41039-1. doi: 10.1007/978-3-658-41039-1_21. URL https://doi.org/10.1007/978-3-658-41039-1_21.
- [54] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL <http://arxiv.org/abs/1907.11692>.
- [55] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL <http://arxiv.org/abs/1907.11692>.
- [56] Justus Mattern, Yu Qiao, Elma Kerz, Daniel Wiechmann, and Markus Strohmaier. FANG-COVID: A new large-scale benchmark dataset for fake news detection in German. In Rami Aly, Christos Christodoulopoulos, Oana Cocarascu, Zhijiang Guo, Arpit Mittal, Michael Schlichtkrull, James Thorne, and Andreas Vlachos, editors, *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, pages 78–91, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.fever-1.9. URL <https://aclanthology.org/2021.fever-1.9/>.

- [57] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL <https://arxiv.org/abs/1301.3781>.
- [58] Sima Siami Namini, Neda Tavakoli, and Akbar Siami Namin. A comparative analysis of forecasting financial time series using arima, lstm, and bilstm, 2019. URL <https://arxiv.org/abs/1911.09512>.
- [59] Qiong Nan, Danding Wang, Yongchun Zhu, Qiang Sheng, Yuhui Shi, Juan Cao, and Jintao Li. Improving fake news detection of influential domain via domain- and instance-level transfer, 2022. URL <https://arxiv.org/abs/2209.08902>.
- [60] William S Noble. What is a support vector machine? *Nature Biotechnology*, 24(12):1565–1567, 2006. doi: 10.1038/nbt1206-1565. URL <https://doi.org/10.1038/nbt1206-1565>.
- [61] Isaac Nti, Owusu Nyarko-Boateng, and Justice Aning. Performance of machine learning algorithms with different k values in k-fold cross-validation. *International Journal of Information Technology and Computer Science*, 6:61–71, 12 2021. doi: 10.5815/ijitcs.2021.06.05.
- [62] Tim Osing. *Perspektiven des Onlinejournalismus*, pages 235–247. Springer Fachmedien Wiesbaden, Wiesbaden, 2022. ISBN 978-3-658-39105-8. doi: 10.1007/978-3-658-39105-8_18. URL https://doi.org/10.1007/978-3-658-39105-8_18.
- [63] M. Parmar and A. Tiwari. Enhancing text classification performance using stacking ensemble method with tf-idf feature extraction. In *5th International Conference on Artificial Intelligence and Data Science*, page n/a. IEEE, 2024. URL <https://ieeexplore.ieee.org/abstract/document/10493890/>.
- [64] Peltarion. How to get meaning from text with language model BERT | AI Explained. <https://www.youtube.com/watch?v=-9vVhYEXeyQ>, September 2020. YouTube video.
- [65] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. ACL Anthology, 2014. URL <http://www.aclweb.org/anthology/D14-1162>.
- [66] Aleksandar Petrovic, Jasmina Perisic, Luka Jovanovic, Miodrag Zivkovic, Milos Antonijevic, and Nebojsa Bacanin. Natural language processing approach for fake news

- detection using metaheuristics optimized extreme gradient boosting. In *2024 IEEE 3rd World Conference on Applied Intelligence and Computing (AIC)*, pages 252–257, 2024. doi: 10.1109/AIC61668.2024.10731062.
- [67] Sokleng Prom, Panharith Sun, Neil Ian Cadungog-Uy, Sa Math, and Tharoeun Thap. A bilstm-based sentiment analysis scheme for khmer nlp in time-series data. In *2024 International Conference on Information Science and Communications Technologies (ICISCT)*, pages 133–138, 2024. doi: 10.1109/ICISCT64202.2024.10956585.
- [68] Shahzad Qaiser and Ramsha Ali. Text mining: Use of tf-idf to examine the relevance of words to documents. *International Journal of Computer Applications*, 181, 07 2018. doi: 10.5120/ijca2018917395.
- [69] Nishant Rai, Deepika Kumar, Naman Kaushik, Chandan Raj, and Ahad Ali. Fake news classification using transformer based enhanced lstm and bert. *International Journal of Cognitive Computing in Engineering*, 3:98–105, 2022. ISSN 2666-3074. doi: <https://doi.org/10.1016/j.ijcce.2022.03.003>. URL <https://www.sciencedirect.com/science/article/pii/S2666307422000092>.
- [70] Oona Rainio, Jarmo Teuho, and Riku Klén. Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, 14(1):6086, 2024. doi: 10.1038/s41598-024-56706-x. URL <https://doi.org/10.1038/s41598-024-56706-x>.
- [71] Muhammad Sabir, Talha Khan, and Muhammad Azam. A comparative study of traditional and hybrid models for text classification. *A Comparative Study of Traditional and Hybrid Models for Text Classification*, 03 2025.
- [72] Dipanjan Sarkar. A practitioner’s guide to natural language processing (part i) – processing & understanding text. <https://towardsdatascience.com/a-practitioners-guide-to-natural-language-processing-part-i-processing-understanding-text-9f4abfd13e72/>, 2018. Zugriff am 12. Mai 2025.
- [73] Philipp Schaer. *C 9 Sprachmodelle und neuronale Netze im Information Retrieval*, pages 455–466. De Gruyter Saur, Berlin, Boston, 2023. ISBN 9783110769043. doi: doi:10.1515/9783110769043-039. URL <https://doi.org/10.1515/9783110769043-039>.
- [74] Mareike Schumacher. Methodenbeitrag: word2vec. *forTEXT*, 1(10), 2024. doi: 10.48694/fortext.3815. URL <https://fortext.net/routinen/methoden/>

- [word2vec-1](#). Erstveröffentlichung: 19.04.2023 auf forttext.net, offizielle Publikation am 30.10.2024.
- [75] Scikit-learn contributors. K-nearest neighbors classifier — hyperparameters overview. <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>, 2024. Zugriff am 15. Mai 2025.
 - [76] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909, 2015. URL <http://arxiv.org/abs/1508.07909>.
 - [77] Upasna Sharma and Jaswinder Singh. A comprehensive overview of fake news detection on social networks. *Social Network Analysis and Mining*, 14(1):120, 2024. doi: 10.1007/s13278-024-01280-3. URL <https://doi.org/10.1007/s13278-024-01280-3>.
 - [78] Shui-Long Shen, Pierre Guy A. Njock, Annan Zhou, and Hai-Min Lyu. Dynamic prediction of jet grouted column diameter in soft soil using bilstm deep learning. *Acta Geotechnica*, 16, 01 2021. doi: 10.1007/s11440-020-01005-8.
 - [79] Sandler Simone, Krauss Oliver, Diesenreiter Clara, and Stöckl Andreas. Detecting fake news and performing quality ranking of german newspapers using machine learning. In *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, pages 1–5, 2022. doi: 10.1109/ICECCME55909.2022.9987851.
 - [80] M. Sudhakar and K.P. Kaliyamurthie. Detection of fake news from social media using support vector machine learning algorithms. *Measurement: Sensors*, 32: 101028, 2024. ISSN 2665-9174. doi: <https://doi.org/10.1016/j.measen.2024.101028>. URL <https://www.sciencedirect.com/science/article/pii/S2665917424000047>.
 - [81] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification?, 2020. URL <https://arxiv.org/abs/1905.05583>.
 - [82] M. Umar, H. D. Abubakar, and M. A. Bakale. Sentiment classification: Review of text vectorization methods: Bag of words, tf-idf, word2vec and doc2vec. *SLU Journal of Science and Technology*, 2022. URL https://www.academia.edu/download/107531976/Mahmood_and_Haisal_pub2.pdf.

- [83] Muhammad Umer, Zainab Imtiaz, Saleem Ullah, Arif Mehmood, Gyu Sang Choi, and Byung-Won On. Fake news stance detection using deep learning architecture (cnn lstm). *IEEE Access*, 8:156695–156706, 2020. doi: 10.1109/ACCESS.2020.3019735.
- [84] Rajkumar V and Priyadarshini G. Roberta-lightgbm: A hybrid model of deep fake detection with pre-trained and binary classification. *INFOCOMP Journal of Computer Science*, 23(1), Jul. 2024. URL <https://infocomp.dcc.ufla.br/index.php/infocomp/article/view/3060>.
- [85] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL <https://arxiv.org/abs/1706.03762>.
- [86] Birendra Kumar Verma and Ajay Kumar Yadav. Advancing software vulnerability scoring: A statistical approach with machine learning techniques and gridsearchcv parameter tuning. *SN Computer Science*, 5(5):595, 2024. doi: 10.1007/s42979-024-02942-x. URL <https://doi.org/10.1007/s42979-024-02942-x>.
- [87] Pawan Kumar Verma, Prateek Agrawal, Vishu Madaan, and Radu Prodan. Mcred: multi-modal message credibility for fake news detection using bert and cnn. *Journal of Ambient Intelligence and Humanized Computing*, 14(8):10617–10629, 2023. doi: 10.1007/s12652-022-04338-2. URL <https://doi.org/10.1007/s12652-022-04338-2>.
- [88] Shirui Wang, Wenan Zhou, and Chao Jiang. A survey of word embeddings based on deep learning. *Computing*, 102(3):717–740, 2020. doi: 10.1007/s00607-019-00768-7. URL <https://doi.org/10.1007/s00607-019-00768-7>.
- [89] Yuxiang Wang, Yongheng Zhang, Xuebo Li, and Xinyao Yu. Covid-19 fake news detection using bidirectional encoder representations from transformers based models, 2021. URL <https://arxiv.org/abs/2109.14816>.
- [90] Geoffrey I Webb, Eamonn Keogh, and Risto Miikkulainen. Naïve bayes. *Encyclopedia of machine learning*, 15(1):713–714, 2010.
- [91] Alexander Wettig, Tianyu Gao, Zexuan Zhong, and Danqi Chen. Should you mask 15 percent in masked language modeling, 2023. URL <https://arxiv.org/abs/2202.08005>.

- [92] Harry Zhang. The optimality of naive bayes. *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004*, 2, 01 2004.

A Anhang

A.1 Verwendete Hilfsmittel

In der Tabelle A.1 sind die im Rahmen der Bearbeitung des Themas der Bachelorarbeit verwendeten Werkzeuge und Hilfsmittel aufgelistet.

Tabelle A.1: Verwendete Hilfsmittel und Werkzeuge

Tool	Verwendung
L <small>A</small> T <small>E</small> X	Textsatz- und Layout-Werkzeug verwendet zur Erstellung dieses Dokuments

A.2 Abbildungen

A.3 Tabellen

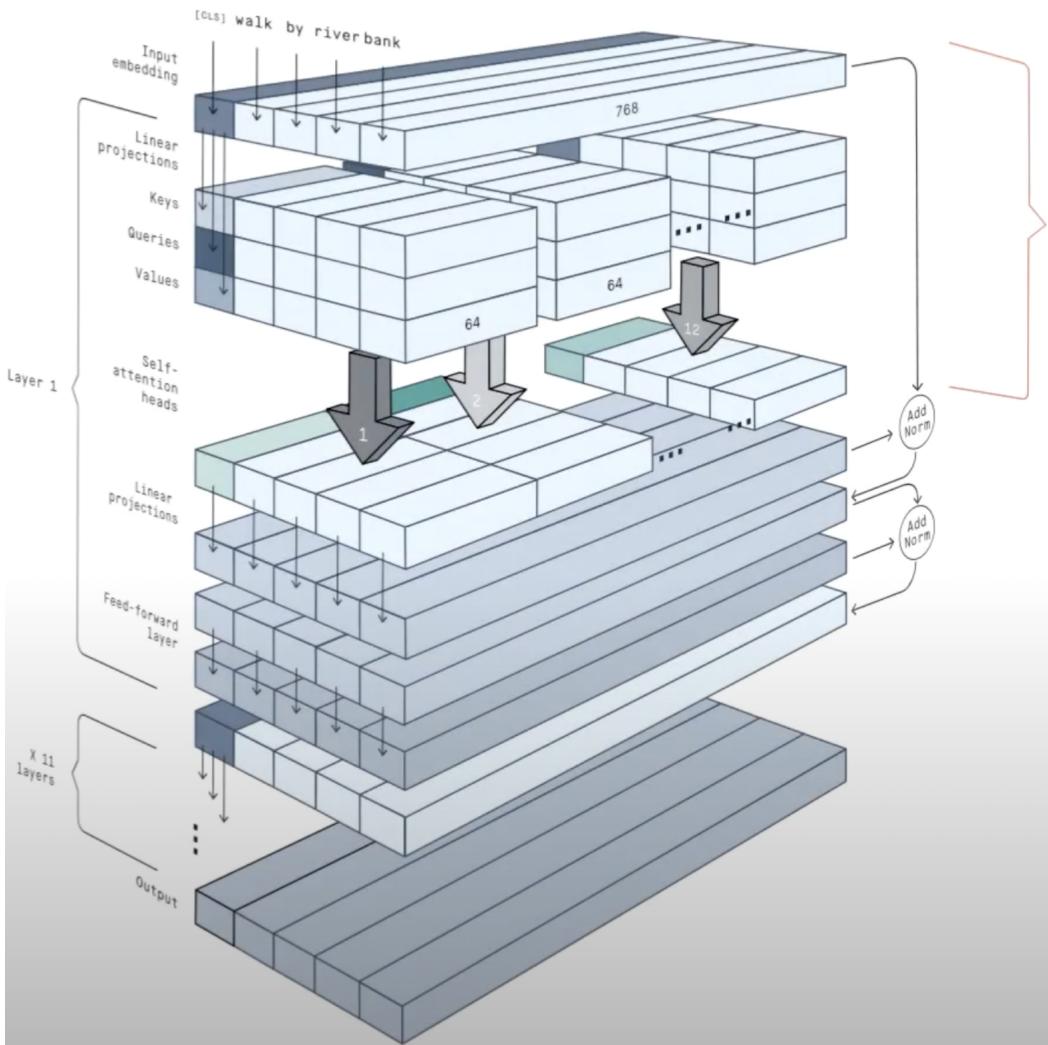


Abbildung A.1: Architektur des BERT Modells [64]

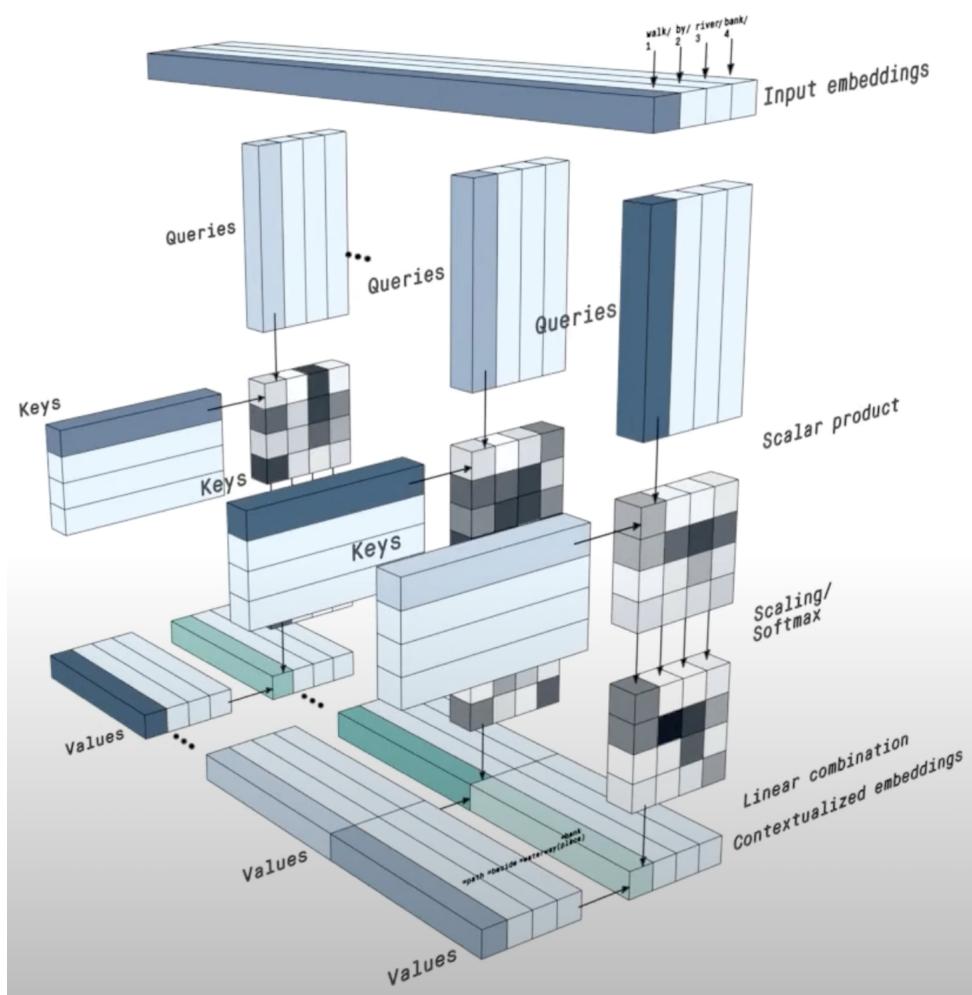


Abbildung A.2: Multi-Head Attention [64]

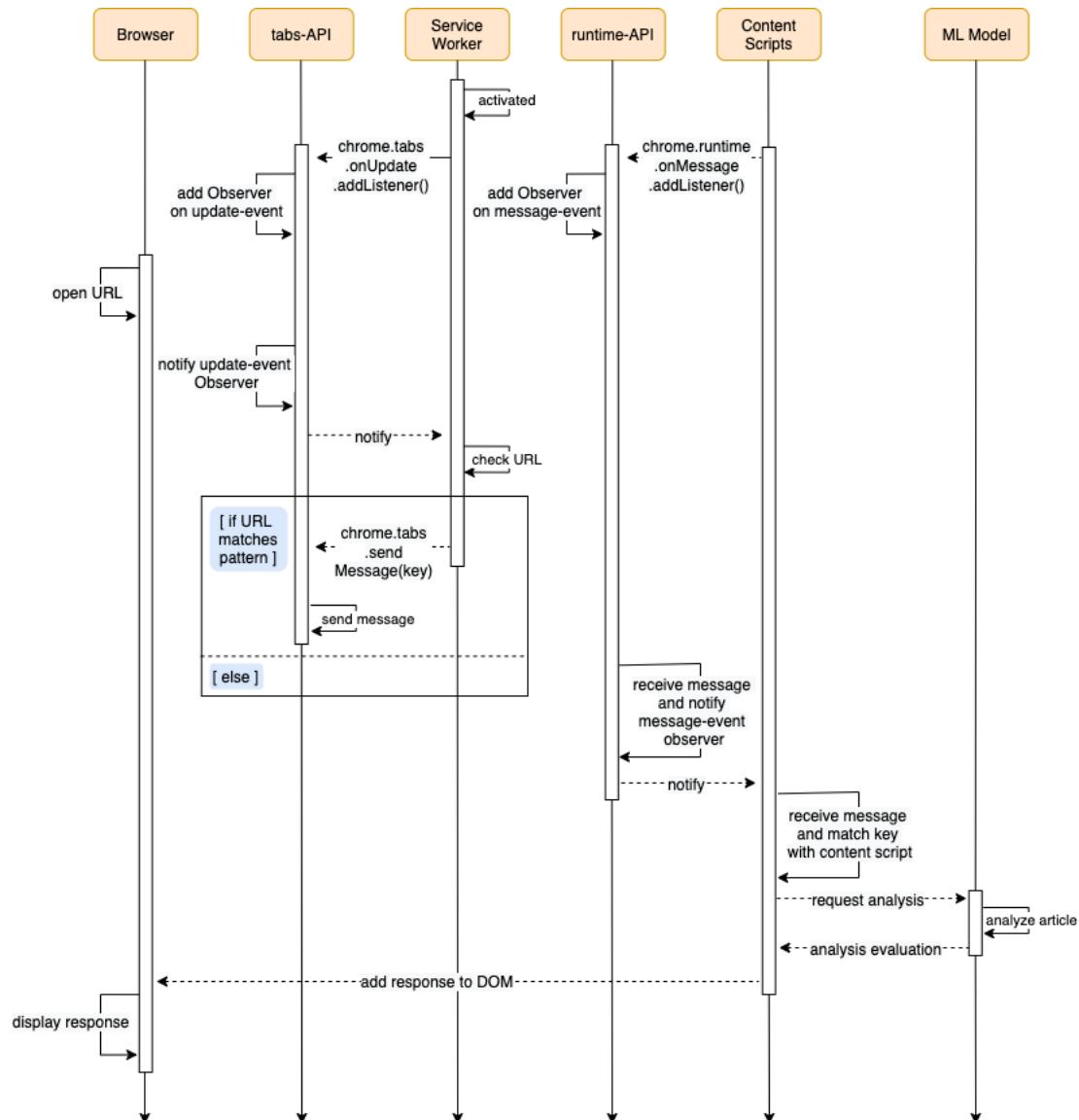


Abbildung A.3: Sequenzdiagramm Webagent

Kriterium	Chrome Extension	Userscript (Tampermonkey)	Proxy-Server	Scraper + Plattform
DOM-Zugriff beim Nutzer	Ja	Ja	Nein	Nein
Einbindung auf bild.de direkt	Ja	Ja	Ja (indirekt)	Nein
Installation durch Nutzer	Mittel	Einfach	Nicht erforderlich	Nicht erforderlich
Komplexität der Umsetzung	Mittel	Gering	Hoch	Mittel
Wartbarkeit & Updates	Gut	Gut	Aufwändig	Mittel
Performance beim Nutzer	Hoch	Hoch	Hoch	Hoch
Skalierbarkeit	Hoch	Eingeschränkt	Mittel	Hoch
Für öffentliche Verbreitung geeignet	Ja	Eingeschränkt	Eingeschränkt	Ja
API-Nutzung zur Fake-Erkennung	Ja	Ja	Ja	Ja
Entwicklerkontrolle über UI	Hoch	Mittel	Hoch	Mittel

Tabelle A.2: Vergleich möglicher Technologien für den Webagenten

Merkmal	BERT Base	RoBERTa Base	RoBERTa Large	XLM- RoBERTa Base	XLM- RoBERTa Large
Hidden Size	768	768	1024	768	1024
Anzahl Layer	12	12	24	12	24
Anzahl Attention Heads	12	12	16	12	16
Vocab Size	30,522	50,265	50,265	250,002	250,002
Spezialisiert auf Sprachumfang	Masked LM Englisch	Masked LM Englisch	Masked LM Englisch	Multilinguale Masked LM Multilingual	Multilinguale Masked LM Multilingual

Tabelle A.3: Vergleich der verschiedenen BERT- und RoBERTa-Modelle

Parameter	Wert	Beschreibung
Anzahl Trainingsepochen	5	Das gesamte Trainingsset wird fünfmal vollständig durchlaufen.
Batch-Größe	32	Anzahl der Beispiele, die gleichzeitig in einem Schritt verarbeitet werden.
Lernrate	$2e-5$	Bestimmt die Schrittweite der Modellaktualisierung bei jedem Optimierungsschritt.
Optimierungsverfahren	AdamW	Variante des Adam-Optimierers mit Weight Decay, automatisch in Hugging Face integriert.
Gewichtsabnahme (Weight Decay)	0.01	Reguliert große Gewichtswerte, um Überanpassung zu vermeiden.
Lernraten-Scheduler	Linear, 10% Warmup	Die Lernrate steigt linear an und wird anschließend schrittweise reduziert.
Kriterium für bestes Modell	F1-Score	Das Modell mit dem besten F1-Score auf den Validierungsdaten wird gespeichert.
Hardware-Beschleunigung	fp16 aktiviert	Angepasst auf A100 GPU zur Beschleunigung des Trainings.

Tabelle A.4: Überblick über die gewählten Hyperparameter der Transformer-Modelle

Parameter	Wert	Beschreibung
Ziel (objective)	binary	Binäre Klassifikation (z. B. „echt“ oder „fa- ke“).
Metrik (metric)	binary_logloss	Verlustfunktion zur Bewertung der Mo- dellgüte während des Trainings.
Boosting-Typ (boosting_ty- pe)	gbdt	Verwendung von Gradient Boosted Deci- sion Trees als Lernverfahren.
Prozessor-Kerne (n_jobs)	-1	Nutzt alle verfügbaren CPU-Kerne für paralleles Training.
Lernrate (learning_ra- te)	0.0865	Schrittweite beim Anpassen der Modellge- wichte.
Anzahl Blätter (num_leaves)	63	Maximale Anzahl von Blättern pro Ent- scheidungsbaum.
Maximale Tiefe (max_depth)	20	Begrenzt die Tiefe der Bäume zur Kontrol- le der Komplexität.
Min. Samp- les pro Blatt (min_child_- samples)	80	Minimale Anzahl an Trainingsbeispielen pro Blatt.
Subsample-Rate (subsample)	0.9818	Anteil der Trainingsdaten, der zufällig pro Baum verwendet wird.
Merkmalsauswahl pro Baum (colsample_- bytree)	0.9684	Anteil der Merkmale, die pro Baum zufäl- lig ausgewählt werden.
L_1 - Regularisierung (reg_alpha)	0.2989	Bestraft große Gewichtswerte zur Förde- rung einfacher Modelle.
L_2 - Regularisierung (reg_lambda)	0.4609	Stabilisiert das Modell durch Bestrafung großer Gewichtssummen.
Zufallsstart (random_- state)	42	Sichert Reproduzierbarkeit der Ergebnis- se.
Anzahl Bäume (n_estima- tors)	2000	Maximale Anzahl an Bäumen; Early Stop- ping begrenzt effektiv.

Tabelle A.5: Überblick über die gewählten Hyperparameter des LightGBM-Modells

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