HAW HAMBURG

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

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

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

Betreuender Prüfer: Prof. Dr. Stefan Sarstedt
Zweitgutachterin: Prof. Dr. Marina Tropmann-Frick

Eingereicht am: 03.07.2025

Kristoffer Schaaf

Thema der Arbeit

Entwicklung einer Software zur Erkennung von Fake News auf Nachrichtenportalen

Stichworte

Machinelles Lernen, Fake News Erkennung, Textklassifikation, Natural Language Processing, Transformer, BERT, RoBERTa, LightGBM, Chrome-Extension

Kurzzusammenfassung

Die Bachelorarbeit dokumentiert die Entwicklung einer Software zur Erkennung von Fake News auf Nachrichtenportalen mithilfe von Methoden des maschinellen Lernens und modernen Ansätzen des Natural Language Processings. Ziel ist die automatisierte Klassifikation von Nachrichtenartikeln als echt oder gefälscht auf Basis semantischer und stilistischer Merkmale. In dieser Arbeit erfolgt ein Fine-Tuning verschiedener Transformer-Modelle, deren Embeddings anschließend als Eingabe für LightGBM-Modelle dienen, um die Klassifikation von Fake News effizient umzusetzen. Die Lösung wird mithilfe einer Chrome-Erweiterung für drei deutsche Nachrichtenportale implementiert, die in ihrer Gesamtheit politisch divers sind (BILD, taz, Der Spiegel). Evaluiert werden die Ergebnisse auf Basis eines eigens zusammengestellten Datensatzes. Die Arbeit erläutert Vorverarbeitungsschritte, Modellarchitekturen, Trainingsverfahren und zeigt durch systematische Evaluation die Effektivität der entwickelten hybriden Klassifikationslösung auf.

Kristoffer Schaaf

Title of Thesis

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

Keywords

machine learning, fake news detection, text classification, natural language processing, transformer, BERT, RoBERTa, LightGBM, Chrome extension

Abstract

The bachelor's thesis documents the development of software for detecting fake news on news portals using machine learning methods and modern Natural Language Processing approaches. The goal is the automated classification of news articles as real or fake based on semantic and stylistic features. In this work, various transformer models are fine-tuned, and their embeddings are subsequently used as input for LightGBM models to efficiently implement fake news classification. The solution is implemented for three politically diverse German news portals (BILD, taz, Der Spiegel) using a Chrome Extension. The results are evaluated using a custom-compiled dataset. The thesis explains preprocessing steps, model architectures, and training procedures and demonstrates the effectiveness of the developed hybrid classification solution through systematic evaluation.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

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

„They are eating the cats (dt: Die essen Katzen)“. Dieser Ausspruch des amerikanischen Präsidenten Donald Trump ist wohl einer der prägendsten Momente des US-amerikanischen Präsidentschaftswahlkampfs 2024. Wie die beiden Quellen BBC News und The Guardian belegen, behauptete Trump während einer Fernsehdebatte am 10. September, dass Einwanderer in Springfield, Ohio, speziell haitianische Zuwanderer, Hunde und Katzen der Einwohner „essen“ würden [45, 64]. Moderator David Muir reagierte umgehend, indem er erklärte, dass die Stadtverwaltung keine glaubhaften Hinweise auf derartige Vorfälle habe. Wie sich später herausstellte beruhten diese Behauptung auf Gerüchten, die ursprünglich aus einem Facebook-Post in einer privaten Gruppe stammten. Dort hieß es, jemand habe eine Katze gesehen, die angeblich für den Verzehr aufgehängt wurde. Trumps Behauptung beruhten auf widerlegten Gerüchten aus sozialen Medien. Offizielle Stellen aus Springfield, darunter der City Manager, der Bürgermeister, die Polizei und der Gouverneur Ohios, wiesen die Behauptung klar zurück und erklärten, es gebe keine belastbaren Beweise für Gewalt gegen Haustiere durch haitianische Migranten. Trotzdem verbreiteten prominente Vertreter der Republikanischen Partei wie Trumps Vize-Präsident J.D. Vance, Senator Ted Cruz und konservative Influencer die Geschichte weiter [49]. Das führte schlussendlich wohl zu Bombendrohungen und Schulschließungen in Springfield [62]. Trumps Behauptung war falsch und parteiisch motiviert, basierte auf Gerüchten, die vor Ort widerlegt wurden, und bildet damit ein Beispiel dafür, wie Desinformation gezielt in gesellschaftlichen Debatten eingesetzt wird.

1.1.1 Entstehung und Motivation

Das Verbreiten von Desinformationen ist allgegenwärtig. Es geschieht in Familien Chatgruppen, in Klassenzimmern oder eben auf offener Bühne im US-amerikanischen Präsidentschaftswahlkampf. Es ist dabei kein Phänomen des 21. Jahrhunderts. Belege reichen beispielsweise bis ins Jahr 44 v. Chr zurück [9]. Auch während des amerikanischen Unabhängigkeitskrieges im Jahr 1782 wurde die gezielte Verbreitung falscher Informationen als politisches Mittel genutzt. So verfasste Benjamin Franklin einen Brief, der angeblich von Captain Samuel Gerrish stammte. In diesem wurde über Grausamkeiten britischer Soldaten und ihrer Verbündeten berichtet [11]. Ziel dieser Fälschung war es, die britische Krone zu diskreditieren und die öffentliche Meinung zugunsten der amerikanischen Unabhängigkeitsbewegung zu beeinflussen [83].

Der Begriff *Fake News* entwickelte sich im 21. Jahrhundert von der reinen Beschreibung von Desinformation zum politischen Kampfbegriff [6, 15].

Fake News greifen zentral in den Prozess gesellschaftlicher Meinungsbildung und dienen dabei zwei Motivationen: Einerseits soll dieser Meinungsbildungsprozess mit Fake News im Sinne einer Propaganda in eine gewünschte Richtung gelenkt werden [15]. Dies kann gesellschaftliche Stimmungen, aber auch Wahlen beeinflussen, wie das Beispiel Donald Trumps zeigt. Fake News greifen damit den Kern von Demokratie an. Andererseits dienen Fake News einem ökonomischen Interesse. Fake News können beispielsweise Artikel in Online-Zeitungen besonders interessant wirken lassen und deren Reichweite steigern. 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 [9]. Beide Motive untermauern, weshalb das Erkennen von Fakes News auf Nachrichtenportalen und das entsprechende Einsetzen von passenden Instrumenten für eine mündige, demokratische Gesellschaft hohe Relevanz haben.

1.1.2 Definition und Aufbau

Der Begriff Fake News ist weder rechtlich noch wissenschaftlich einheitlich definiert, wird aber in politischen und medialen Kontexten intensiv diskutiert. Zur Einordnung sollen zwei maßgebliche Perspektiven gegenübergestellt werden. Eine aus dem deutschen Bildungsbereich und eine internationale Einordnung der UNESCO:

- **Bundeszentrale für politische Bildung (bpb):** „Fake News“ sind laut bpb „gefälschte Nachrichten“, die mit „reißerischen Schlagzeilen, gefälschten Bildern und Behauptungen [...] Lügen und Propaganda verbreiten“. Ziel sei es, gezielt zu täuschen und den Eindruck echter journalistischer Berichterstattung zu erwecken [39].
- **United Nations Educational, Scientific and Cultural Organization (UNESCO):** Die UNESCO unterscheidet klar zwischen Falschinformationen (unwahre Inhalte, deren Urheberinnen selbst glauben, dass sie wahr seien) und Desinformationen, die „falsch sind und von denen die verbreitende Person weiß, dass sie falsch sind“. Letztere sind gezielt verbreitete Lügen, die darauf abzielen, Rezipientinnen durch böswillige Akteure aktiv zu täuschen [90].

Die bpb fokussiert stärker auf die äußere Erscheinung (z.B. reißerische Gestaltung) und den Propaganda-Charakter von Fake News, während die UNESCO eine analytische Unterscheidung zwischen intentionell irreführenden und unabsichtlich falschen Inhalten einführt.

Als Arbeitsdefinition wird mit folgender Definition fortgefahren: Fake News sind gezielt verbreitete Falsch- und Desinformationen, die unter dem Anschein seriöser Nachrichtenformate gestaltet sind. Ziel ist es, durch bewusste Täuschung die Wahrnehmung, das Verhalten oder die Meinungsbildung der Rezipientinnen zu manipulieren. Sei es aus politischen, ideologischen oder wirtschaftlichen Motiven. Charakteristisch sind täuschend echte Gestaltung, emotionale Sprache und eine bewusste Imitation journalistischer Formate.

Fake News lassen sich nach [83] in sechs Hauptkategorien unterteilen, die sich hinsichtlich ihrer Absichten, Erscheinungsformen und Wirkmechanismen unterscheiden: Satire, Clickbait, Gerüchte, Stance News, Propaganda und Large Scale Hoaxes. Diese Typologie basiert auf einer systematischen Metaanalyse von über 150 wissenschaftlichen Arbeiten.

- **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 [6].

1.1.3 Verbreitung von Fake News

Der Austausch von Nachrichten ist elementarer Bestandteil des Meinungsbildungsprozesses in einer demokratischen Gesellschaft. Dieser Prozess entfaltet in sozialen Medien eine besondere Dynamik. Hier 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 [9].

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 [47].

1.1.4 Kosumenten

Laut [47] sind folgende Gruppen die größten Konsumenten von Fake News:

- **Geringe Bildung oder digitale Kompetenz:** Personen mit niedriger formaler Bildung oder unzureichenden digitalen Fähigkeiten sind anfälliger für Falschinformationen.
- **Persönliche Nähe zur Informationsquelle:** Informationen von Personen, denen man persönlich nahe steht oder vertraut, werden, unabhängig vom Wahrheitsgehalt, eher geglaubt.
- **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 [9].

1.1.5 Indikatoren zum Erkennen von Fake News

Das Erkennen von Fake News ist gerade deshalb problematisch, da diese erst erkannt werden können, nachdem sie erstellt und im Internet verbreitet wurden. [83] Gerade im Bereich der sozialen Medien gibt es aber relativ zuverlässige Indikatoren, die Fake News nach der Erstellung als solche zu enttarnen [42]:

- **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 Automatisierte Erkennung von Fake News

Die TU Darmstadt stellt in der Arbeit [42] das Browser Pluging *Trusty Tweet* vor. Dieses Tool unterstützt Benutzer*innen bei der Bewertung von Tweets auf Twitter, indem es politisch neutrale und intuitive Warnungen anzeigt, ohne Reaktanz zu erzeugen. In [85] wird zudem gezeigt, wie maschinelle Lernverfahren zur automatischen Erkennung von Fake News und zur Bewertung von Nachrichtenquellen eingesetzt werden können. Motiviert durch diese Vorarbeiten wird in dieser Arbeit die Entwicklung eines weiteren Tools zur Erkennung von Fake News entwickelt und dokumentiert. Dieses Tool soll wie auch das Browser Plugin TrustyTweet eine Unterstützung zum Erkennen von Fake News anbieten. Ziel ist es, das Tool nicht wie TrustyTweet auf Twitter einzusetzen, sondern auf verschiedenen Nachrichtenportalen. Um eine politisch möglichst breites Spektrum zu decken, wird das Tool für drei verschiedene, in ihrer Gesamtheit diverse, Nachrichtenportalen implementiert.

1.3 Wahl der Nachrichtenportale

Im Paper der University of Applied Sciences Upper Austria [85] wird die Qualität verschiedener deutscher Nachrichtenportale mit Machine Learning Modellen getestet. Das Ergebnis zeigt, dass die Zeitungen Spiegel, Die Zeit sowie Süddeutsche die glaubwürdigsten Portale sind. Express, BZ-Berlin und Bild sind die 'schlechtesten', da sie am meisten Fake News verbreiten. Die Arbeiten [44, 59] und [70] belegen, dass die Kombination von BILD, taz und Der Spiegel eine politisch breites Spektrum abbilden.

- **BILD:** Boulevardesk, populistisch, konservativ

  Die BILD-Zeitung gilt als stark meinungsgetriebenes 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.

Die entwickelte Anwendung wird für diese drei Nachrichtenportale implementiert.

1.4 Aufbau der Arbeit

Diese Arbeit gliedert sich in insgesamt acht Kapitel. Kapitel 2 stellt verschiedene Ansätze des maschinellen Lernens und Deep Learning vor, wobei insbesondere klassische Modelle, neuronale Netze und moderne Transformer Architekturen erläutert werden. Darauf aufbauend werden die verwendeten Metriken zur Modellbewertung erläutert und hybride Modellansätze und verwandte Arbeiten gezeigt. Kapitel 3 beschreibt die Auswahl und Eigenschaften relevanter Datensätze sowie die Kriterien, nach denen der finale Datensatz

zusammengestellt wurde. In Kapitel 4 wird die konzeptionelle Planung der Softwarelösung vorgestellt, während Kapitel 5 die technische Umsetzung des Prototyps detailliert erläutert. Nachfolgend sind in Kapitel 6 die Evaluationsergebnisse, Vergleiche zwischen den verschiedenen Modellen und eine Diskussion über deren Leistungsfähigkeit präsentiert. Abschließend fasst Kapitel 7 die wesentlichen Erkenntnisse zusammen und Kapitel 8 gibt einen Ausblick auf mögliche Erweiterungen und zukünftige Forschungsvorhaben.

2 Natural Language Processing

Natural Language Processing (NLP) ist ein Teilgebiet der Künstlichen Intelligenz, das sich mit der automatisierten Verarbeitung und dem Verständnis natürlicher Sprache durch Computer beschäftigt. In der Fake-News-Erkennung spielt NLP eine zentrale Rolle, da Fake News oft sprachliche Merkmale und manipulative Formulierungen enthalten, die sich mit NLP-Methoden analysieren lassen. Durch Verfahren wie Textklassifikation oder semantische Textverarbeitung können relevante Muster erkannt werden, um zwischen glaubwürdigen und manipulierten Inhalten zu unterscheiden.

Im Bereich des NLPs kommen verschiedene Ansätze zum Einsatz. Darunter sind klassische Machine-Learning-Modelle, Deep-Learning-Methoden sowie moderne Transformer-Architekturen, die den aktuellen Stand der Technik repräsentieren.

2.1 Machine Learning

Im Folgenden werden verschiedene Datenvorverarbeitungsverfahren und Modelle des maschinellen Lernens vorgestellt.

2.1.1 Textbereinigung und Vorverarbeitung

Ein zentraler Schritt bei der Anwendung von NLP-Methoden, gerade beim maschinellen Lernen, ist die Bereinigung und Vorverarbeitung von Textdaten. Ziel ist es, den Text in eine strukturierte Form zu überführen, die für nachgelagerte Klassifikations- oder Analyseverfahren nutzbar ist.

Folgende Schritte sollten bei der Vorbereitung unstrukturierter Textdaten von Nachrichtenartikeln beachtet werden:

- **Titel und Inhalt der Artikel zusammenfügen [14]**: 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 [14] [78] [7]**: 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 [78] [86] [7]**: Ähnlich wie zum vorherigen Punkt erhöht die durchgehende Kleinschreibung aller Buchstaben die Generalisierung und verhindert somit unnötige Duplikate im Vokabular.
- **Leere Spalten entfernen [86]**: 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) [14]**: 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 [14] [78] [7]**: Wörter wie „der“, „ist“, „und“ tragen wenig zur inhaltlichen Differenzierung bei. Das Entfernen dieser verbessert die semantische Gewichtung relevanter Begriffe [79].
- **Rechtschreibfehler korrigieren [78]**: 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 [14] [78] [7]**: 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 [78]**: 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.

2.1.2 Merkmalextraktion

Nach der Bereinigung und Vorverarbeitung der Textdaten folgt die Extraktion relevanter Merkmale, um die Inhalte in eine mathematisch verwertbare Repräsentation zu überführen. Diese Merkmale bilden die Grundlage für die nachfolgenden maschinellen Lernprozesse, da sie den inhaltlichen und strukturellen Charakter der Texte in numerischer Form abbilden. Je nach gewähltem Verfahren können unterschiedliche sprachliche Eigenschaften erfasst und für Klassifikationsmodelle zugänglich gemacht werden.

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. Die Reihenfolge oder der Kontext der Wörter wird dabei nicht beachtet. Es zählt lediglich das Vorkommen jedes Wortes aus einem festen Vokabular [19].

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 [33].

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 [14]

	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 [14]

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:

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

Dabei ist t das Wort, d das Dokument und D die gesamte Dokumentensammlung [7].

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

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

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.

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

Dabei ist N die Gesamtanzahl der Dokumente in der Matrix und $\text{df}(t)$ die Anzahl der Dokumente, in denen der Begriff t vorkommt [74].

Die in [29] 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:

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

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:

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

- 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

Bag-of-Words (BoW)	TF-IDF
(+) Einfache Implementierung [19]	(+) Berücksichtigt Wortwichtigkeit in gesamter Matrix [33]
(–) Keine Gewichtung — häufige Wörter dominieren	(+) Seltener, aber informativer Inhalt wird stärker gewichtet [23]
(–) Hohe Dimensionalität in Sparse Matrix (jedes Wort bekommt eine separate Dimension) [22]	(+) Gleiches Problem, aber mit informativeren Werten [5]
(–) Ignoriert Wortreihenfolge und Kontext [88]	(+) Gleiches Grundproblem, aber geringfügig bessere Performance [71]
(+) Nützlich für einfache Klassifikatoren	(+) Bessere Klassifikationsergebnisse in Kombination mit SVM oder Logistic Regression [50]

Tabelle 2.1: Vergleich der Vorteile (+) und Nachteile (–) von BoW 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 [14].

In Abbildung 2.2 wird der Vergleich zwischen Machine-Learning-Modellen unter Verwendung von BoW-, TF-IDF- und Hashing-Features gezeigt. Die y-Achse bildet die Genauigkeit der Modelle ab. 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 Werte mit Hashing-Features verbessern konnten.

Die Verbesserung ist aber nur minimal. Bei SVM ist der Wert ohne Hashing bei 0.89 und nach bei 0.90. Bei LR steigt er von 0.87 und 0.88.

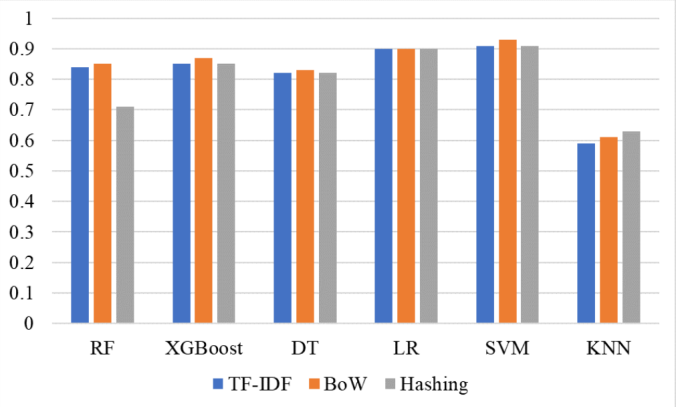

Abbildung 2.2: Vergleich verschiedener Modelle mit BoW, TF-IDF und Hashing [7]

2.1.3 Machine Learning Modelle

Unter anderem kommen folgende Modelle bei der Klassifizierung von Nachrichtenartikeln zum Einsatz.

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 [13].

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, wenn alle Daten zur selben Klasse gehören. Die Entropie $E(S)$ eines Datensatzes S wird wie folgt berechnet:

$$E(S) = -\sum_{i=1}^{c} p_i \log_2 p_i \tag{2.6}$$

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:

$$IG(E) = 1 - \sum_{i=1}^{c} p_i^2 \tag{2.7}$$

Über einen Hyperparameter kann die maximale Tiefe des Baumes festlegt werden. Eine zu große Tiefe kann zu Overfitting führen, da der Baum zu sehr an die Trainingsdaten angepasst wird [7].

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) [32].

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:

$$RFf_i = \frac{\sum_{j \in \text{all trees}} normf_{ij}}{T} \tag{2.8}$$

Dabei ist:

- RFf_i die Gesamtrelevanz der Klasse i im gesamten Wald,
- $normf_{ij}$ die normierte Wichtigkeit des Merkmals i im Baum j,
- T die Gesamtanzahl der Entscheidungsbäume [7].

Wichtige Hyperparameter für dieses Modell sind:

- `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

Eine Support Vector Machine ist ein überwachte Lernalgorithmus, welcher besonders effektiv für Klassifikationsaufgaben ist. Er findet breite Anwendung in Bereichen wie Bioinformatik, Textklassifikation und insbesondere in der Erkennung von Fake News. Das Ziel einer SVM ist es eine Gerade (in 2D), Ebene (in 3D) oder Hyperplane zu finden, die Datenpunkte verschiedener Klassen mit maximalem Abstand (Margin) voneinander trennen (siehe Abbildung 2.3) [68, 14, 78, 51].

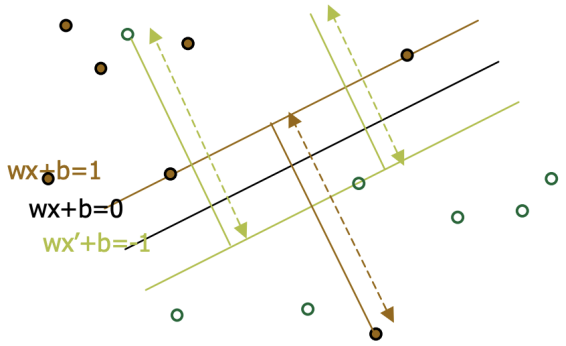

Abbildung 2.3: Darstellung von Hyperplanes [51]

Bei komplexen Datensätzen wird das Problem durch eine Kernel-Funktion in höhere Dimensionen überführt [51].

Gängige Kernel-Funktionen sind:

- Linearkernel: $K(x, x') = x^T x'$
- Polynomial: $K(x, x') = (x^T x' + 1)^d$
- Radial Basis Function (RBF): $K(x, x') = \exp(-\gamma \|x - x'\|^2)$ [51]

[68] zeigt, dass durch geeignete Wahl eines Kernels auch komplex strukturierte Daten erfolgreich klassifiziert werden können und außerdem besonders robust gegenüber Overfitting sind. Selbst bei kleinen Datensätzen mit hoher Merkmalsanzahl. In praktischen Anwendungen, etwa bei der Fake-News-Erkennung, erzielen sie zuverlässige Ergebnisse und sind teilweise sogar konkurrenzfähig gegenüber tieferen neuronalen Netzwerken [14, 78].

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 [32, 7, 86]. 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 [7]. Die Tabelle 2.2 zeigt die Aktivierungsfunktionen, mit deren entsprechend benötigten Zielvariablen.

Wie in [7] beschrieben, sind LR-Modelle einfach aufgebaut und liefern Wahrscheinlichkeiten, die direkt interpretierbar sind. Darüber hinaus sind sie effizient in der Anwendung, da sie wenig Rechenleistung benötigen und schnell trainiert werden können und durch ihre Flexibilität kann LR sowohl für binäre als auch für multinomiale und ordinale Klassifikationsprobleme eingesetzt werden [86]. Aufgrund dieser Eigenschaften findet die Methode breite Anwendung in verschiedensten Bereichen, etwa in der Medizin oder bei der automatisierten Textklassifikation [7, 32].

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)
Ordinale 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 [52, 43, 1]

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 2.4). 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 [73, 17, 7].

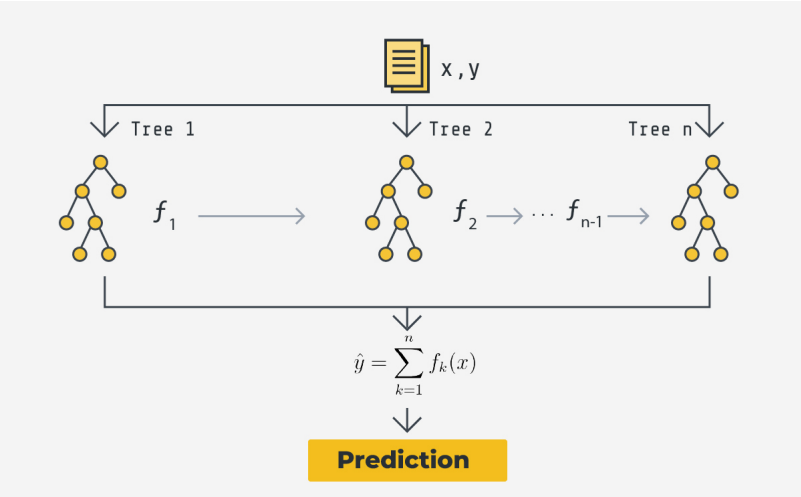

Abbildung 2.4: XGBoost [40]

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.

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 Textmerkmalräumen (z.B. TF-IDF), relevante Merkmale herauszufiltern und gleichzeitig stabile Modelle zu trainieren[17].

LightGBM

Das von [54] 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. *Mutually exclusive* steht dafür, dass sich bestimmte Merkmale gegenseitig ausschließen. Folglich also nicht gleichzeitig einen Wert ungleich Null

annehmen können. 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.

Im Vergleich zu XGBoost, welches die Bäume gleichmäßig pro Ebene aufbaut, wählt LightGBM bei jedem Schritt das Blatt mit dem höchsten Fehler zur Aufspaltung (siehe Abbildung 2.5). Das kann bei LightGBM zu tiefen, asymmetrischen Baumstrukturen führen.

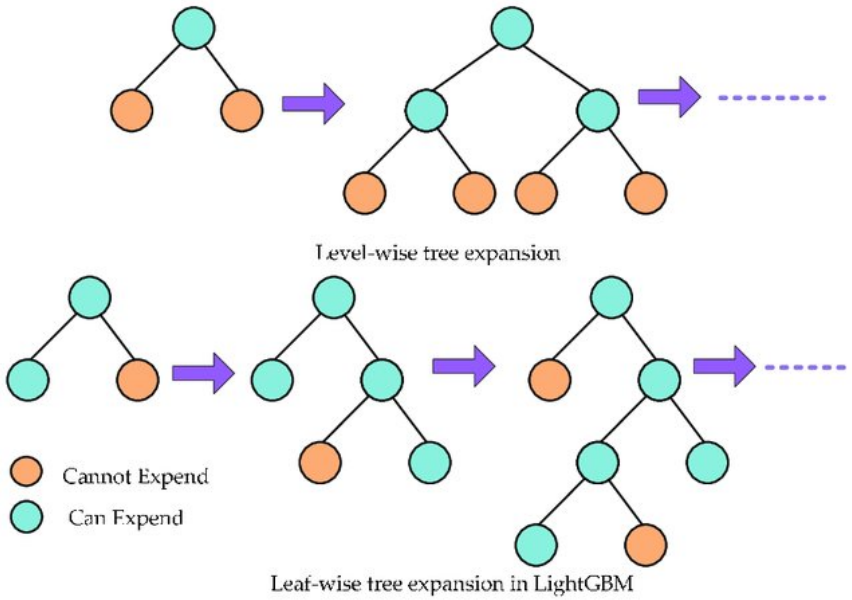

Abbildung 2.5: *level-wise* (XGBoost) vs. *leaf-wise* (LightGBM) [30]

Folgendes Beispiel einer Fake News Klassifizierung zum Verständnis der Funktionsweise von LightGBM:

1. **Startschätzung:** Das Modell beginnt ohne Entscheidungsbäume. Es trifft eine einfache Anfangsschätzung, z.B. dass jeder Artikel mit gleicher Wahrscheinlichkeit echt oder gefälscht ist (z.B. 50:50-Verteilung).

2. **Fehleridentifikation:** Es wird überprüft, welche Artikel falsch klassifiziert wurden. Zum Beispiel: Ein Artikel mit dem Titel „*SCHOCKIEREND: Politiker gesteht*

alles!“ wird fälschlicherweise als echt eingestuft, obwohl es sich um eine Falschmeldung handelt.

3. **Erzeugung eines ersten Entscheidungsbaums:** Basierend auf den erkannten Fehlern wird der erste Baum trainiert, um die fehlerhaften Vorhersagen zu korrigieren. Dabei lernt das Modell etwa, dass reißerische Sprache mit Fake News korreliert.

4. **Anpassung der Vorhersage:** Das Modell passt seine ursprüngliche Schätzung anhand der Regeln des neu erzeugten Baums an. Dieser Baum liefert eine kleine Korrektur, die positiv oder negativ zur bestehenden Vorhersage hinzugefügt wird. So wird die Bewertung des Artikels differenzierter und genauer.

5. **Weitere Iterationen und Baum-Erzeugung:** Nach jeder Anpassung wird erneut überprüft, wo Fehler verbleiben. Ein weiterer Baum wird erzeugt, um diese Fehler zu korrigieren. Dieser Vorgang wird iterativ wiederholt.

6. **Fertiges Modell:** Das finale Modell besteht aus der Summe aller trainierten Bäume. Neue Artikel durchlaufen sämtliche Entscheidungsbäume, deren kombinierte Vorhersagen ergeben eine Wahrscheinlichkeitsaussage.

Sowohl [54] als auch [48] zeigen, dass LightGBM gegenüber XGBoost effizienter trainiert, eine bessere Vorhersage gibt, weniger Merkmale benötigt und kategorische Daten einfacher bearbeiten kann.

2.2 Deep Learning

Deep Learning ist ein Teilbereich des maschinellen Lernens, der auf tiefen neuronalen Netzen basiert und besonders gut für die Verarbeitung großer, unstrukturierter Datenmengen wie Texten geeignet ist. Ein zentrales Element im Bereich des NLP ist dabei die Nutzung von dichten Vektorrepräsentationen von Wörtern (Word Embeddings). Nachstehend werden solche Word Embeddings und zentrale Deep-Learning-Verfahren vorgestellt.

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 [25] 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 [80]. 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 erkennen diese und erstellen für die zwei Kontexte/Wörter zwei verschiedene Vektoren [31].

In Abbildung 2.6 wird jedes Wort eines Korpus mit 6 Wörtern als dreidimensionaler Vektor dargestellt. Ziel eines Word Embedding Verfahrens 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 [10].

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 [10]

Word2Vec

Das Modell Word2vec verwendet ein neuronales Netzwerk und erfasst numerisch die Ähnlichkeiten zwischen Wörtern aufgrund ihrer kontextuellen Merkmale und erstellt darauf aufbauend Embeddings. Am häufigsten wird es zur Analyse der semantischen Verbindungen zwischen Wörtern in einem Textkorpus eingesetzt [81].

Im Beispielsatz 'Mann verhält sich zu Frau wie König zu x.' erkennt Word2vec, dass für x = Königin gilt. Word2Vec löst solche Aufgaben, indem es alle Wörter x' im Gesamtvokabular V ausprobiert und das Wort findet, das folgende Gleichung maximiert [18]:

$$\hat{x} = \underset{x' \in V}{\operatorname{argmax}} \operatorname{sim}(x', \vec{\text{king}} + \vec{\text{woman}} - \vec{\text{man}}) \tag{2.9}$$

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 [81].

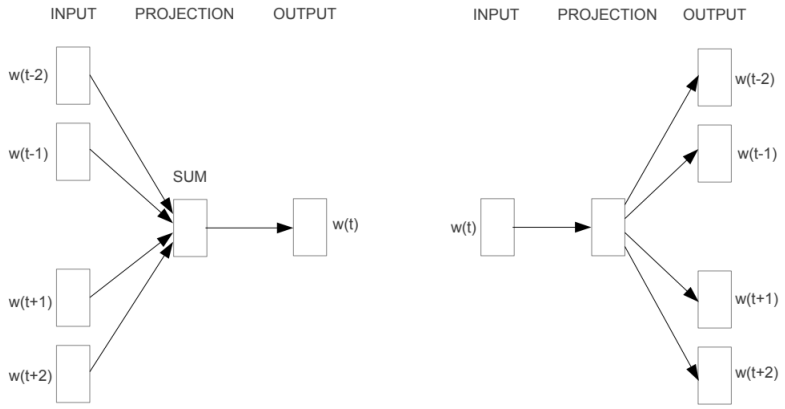

Abbildung 2.7: Vergleich CBOW (links) und Skip-gram (rechts) [65]

GloVe

Word2Vec fokussiert sich auf Informationen aus lokalen Kontextfenstern, 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 [94].

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:

$$P(j \mid i) = \frac{X_{ij}}{X_i} \tag{2.10}$$

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 [72]

Zur Modellierung semantischer Beziehungen vergleicht GloVe Wahrscheinlichkeitsverhältnisse:

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

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

Zur Klassifizierung von Nachrichtenartikeln finden unter anderem folgende Deep-Learning-Modelle Anwendung.

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 [7].

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 [25]. Ein 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. RNNs eignen sich somit besonders für Probleme in der natürlichen Sprachverarbeitung [94], 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, da lange zurückliegende Eingaben nur noch sehr wenig Einfluss auf das Modell nehmen [7].

LSTM

Long Short-Term Memory (LSTM) ist 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 [25].

Ein LSTM-Modell besteht aus mehreren Zellen. Jede dieser Zellen speichert den Zustand des Problems über mehrere Zeitintervalle, während drei Gates den Informationsfluss in die Zelle hinein und wieder heraus regulieren. Das Input-Gate, das Output-Gate und das Forget-Gate [12] (siehe Abbildung 2.9).

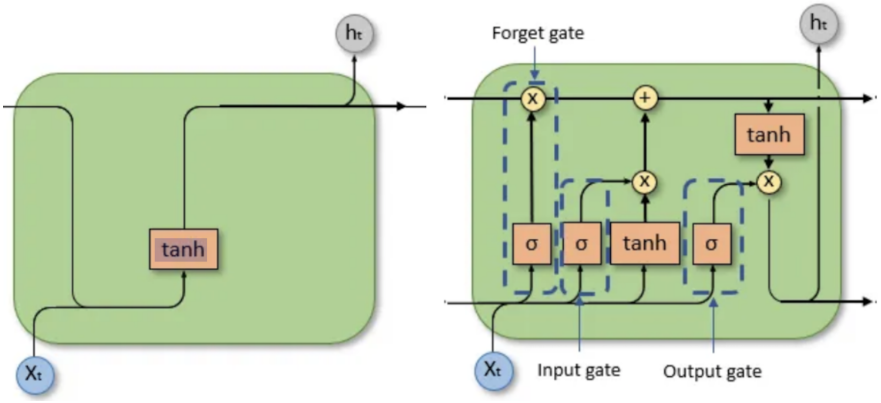

Abbildung 2.9: Vgl. RNN (links) und LSTM (rechts) [2]

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.

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

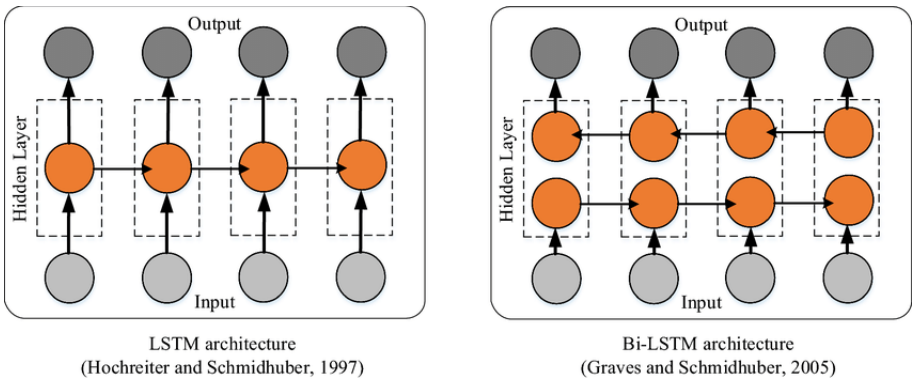

Abbildung 2.10: Vgl. LSTM und BiLSTM [84]

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 [66].

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 Leserichtungen sowohl Informationen aus der Vergangenheit als auch aus der Zukunft zu jedem Zeitpunkt erhalten bleiben können [84].

2.3 Transformer

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 [92].

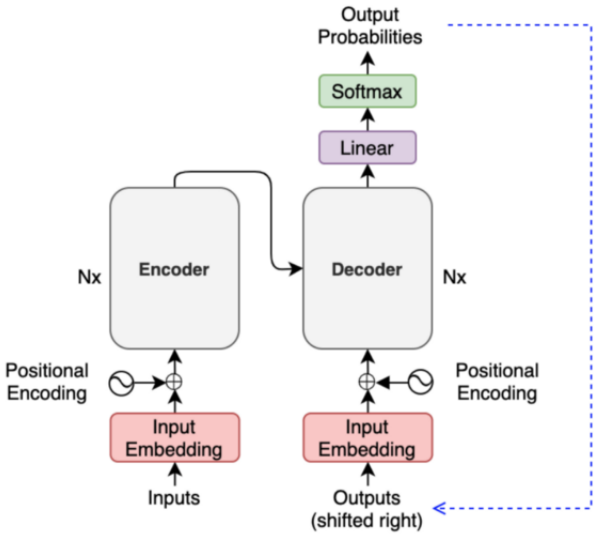

Abbildung 2.11: Eine Übersicht der Transformer Architektur [55] - vereinfacht von [92]

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 [41].

2.3.1 Grundlagen der Transformer Architektur

Self-Attention

In [92] 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 [92] 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:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.11}$$

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 [8].

Im von [92] vorgestelltem Transformer-Modell werden RNNs durch Self-Attention ersetzt.

Hidden Layer

Ein Transformer-Modell [92] besteht aus mehreren Encoder- und/oder Decoder-Schichten. Die erste Schicht erhält den Input und die letzte Schicht gibt den Output aus. Alle anderern Schichten sind versteckt und werden als Hidden Layer bezeichnet.

Im Falle eines Encoding Transformers wie BERT (siehe Kapitel 2.3.2) werden Eingaben durch die verschiedenen Schichten wie folgt verarbeitet [26]:

1. **Tokenisierung der Eingabe**: Jede Eingabe wird in Token umgewandelt. Das Verfahren der Tokenisierung hängt vom verwendeten Transformator Modell ab, z.B. WordPiece beim BERT Modell (siehe Kapitel 2.3.2).
2. **Token-Embedding:** Jeder Token wird anschließend in einen Zahlenvektor umgewandelt. Zusätzlich werden Positions- und Segmentinformationen hinzugefügt.
3. **Verarbeitung durch Hidden Layer:**
    - 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 und bezieht sich auf Wörter davor und danach.

4. **Tiefe Verfeinerung:**
   - Frühe Schichten lernen einfache Beziehungen (z.B. Wortpaare, Syntax).
   - Spätere Schichten modellieren komplexere Abhängigkeiten (z.B. Bedeutung, logische Zusammenhänge).
5. **Ergebnis:**
   - Die letzte Schicht liefert eine kontext-sensitive Repräsentation für jeden Token.
   - In diesem Fall kann diese für Aufgaben wie Klassifikation, Fragebeantwortung oder Named Entity Recognition verwendet werden.

2.3.2 Transformer Modelle

BERT

Bidirectional Encoder Representations from Transformers (BERT) ist ein reiner, für Sprachverständnis optimierter, Encoder-Transformer [27].

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 [25].

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

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

Random initialization

Abbildung 2.12: Bsp. zum MLM [96]

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 [96]. [27] zeigt, wie relevant bidirektionalen Pretrainings für qualitativ hochwertige Sprachrepräsentationen sind.

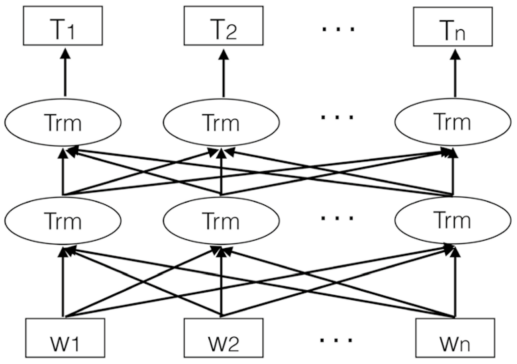

Abbildung 2.13: Bidirektionalität des BERT Modells [94]

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 [35]):

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:

$$\text{Score} = \frac{\text{Häufigkeit des Paares}}{\text{Häufigkeit Teil 1} \times \text{Häufigkeit Teil 2}} \tag{2.12}$$

   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).

Das BERT-Modell umfasst insgesamt ein Vokabular von 30,522 Token.

Das Input Token ergibt sich aus dem Token-, Position- und Segment-Embedding. Das Position-Embedding (Positional Encoding in Abbildung 2.11) stellt hierbei die jeweilige

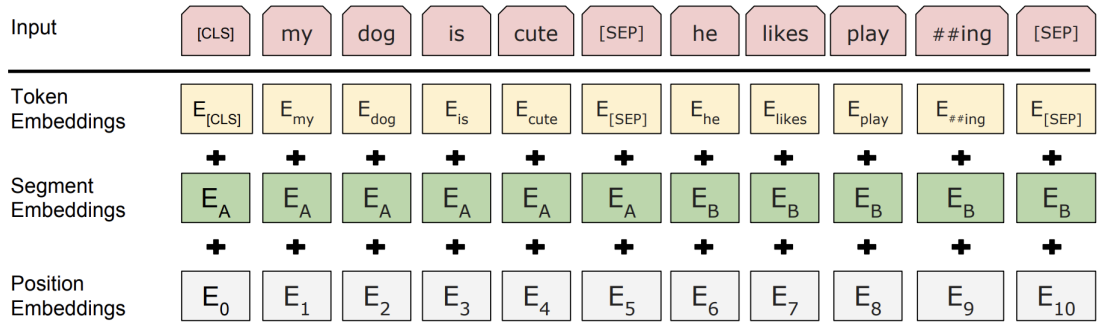

Abbildung 2.14: Zusammensetzung eines Input Tokens im BERT Modell [27]

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) [26].

RoBERTa

Aufgrund der Annahme, dass das in Kapitel 2.3.2 beschriebene BERT Modell nicht ausreichend trainiert wurde, entwickelte [60] 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 [82] 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)` → `"lo"`).
4. Wiederhole Schritt 2-3 für eine feste Anzahl von Merges (z. B. `"lo"` + `"w"` → `"low"` → `[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.

XLM-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 [21].

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 [57]. 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 2.3.2 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.

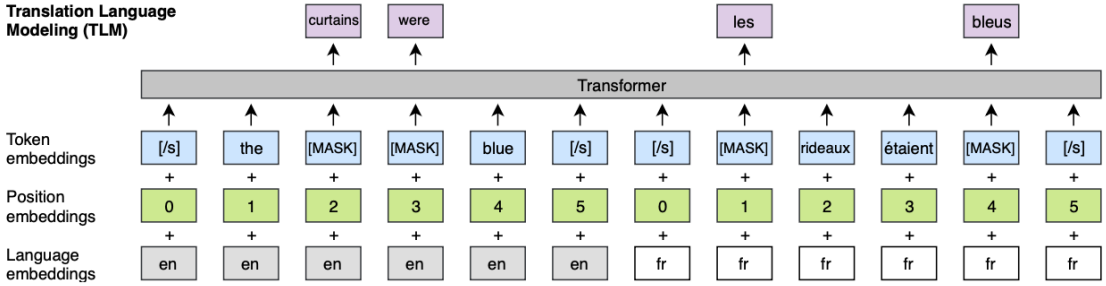

Abbildung 2.15: Auswertung eines bilingualen Satzpaares von TLM [20]

XLM ist ein mehrsprachiges Sprachmodell, das darauf ausgelegt ist, Sprachverständnis über mehrere Sprachen hinweg zu ermöglichen. Vortrainiert wird es mit MLM und zusätzlich TLM. Durch diese Kombination kann das Modell lernen, ähnliche Inhalte in verschiedenen Sprachen miteinander zu verknüpfen [20].

2.4 Metriken

Zur Auswertung der 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' (positive) oder 'falsch' (negative)).

Dabei ergeben sich vier mögliche Ausgänge bei der Modellvorhersage:

- True Positive (TP) - Das Modell hat die positive Klasse richtig 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)

- False Positive (FP) - Das Modell hat die positive Klasse falsch vorhergesagt.

  (Der Artikel, der kein Fake ist, wird als 'falsch' gedeutet)

- False Negative (FN) - Dein Modell hat die negative Klasse falsch vorhergesagt.

  (Der Artikel, der Fake ist, wird als 'wahr' gedeutet)

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

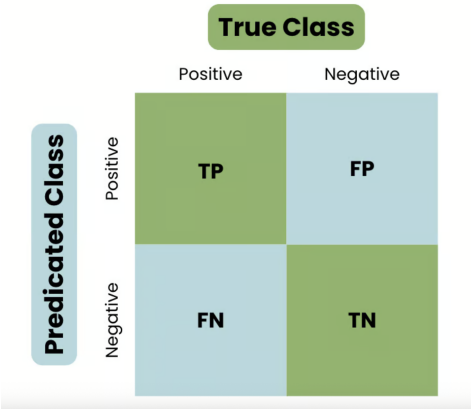

Abbildung 2.16: Konfusionsmatrix [24]

Nach [56, 76] sind die relevantesten Metriken für binäre Klassifikationen Accuracy, Recall (Sensitivity in [76]), Specificity und Precision.

2.4.1 Accuracy

Die Accuracy gibt den Anteil korrekt klassifizierter Instanzen an.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.13}$$

2.4.2 Recall

Der Recall gibt den Anteil korrekt erkannter positiver Fälle unter allen tatsächlich positiven an.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.14}$$

2.4.3 Specificity

Die Specificity gibt den Anteil korrekt erkannter negativer Fälle unter allen tatsächlich negativen an.

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{2.15}$$

2.4.4 Precision

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

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.16}$$

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.

$$\text{F1-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.17}$$

2.5 Hybride Modelle zur Fake News Erkennung

Hybride Modelle kombinieren die Stärken verschiedener Machine- und Deep-Learning-Ansätze, um die Erkennungsgenauigkeit bei Fake News weiter zu verbessern und sowohl semantische als auch stilistische Merkmale effektiv auszuwerten. Im Folgenden werden verschiedene hybride Ansätze zur Fake News Erkennung vorgestellt.

2.5.1 CNN und LSTM mit PCA

In [89] 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 [89]

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 [14] 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 [14]

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 Falschinhalten und ca. 9000 echten Artikeln und ein erweiteter, 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 Datensätz mit dem Keras Embedding bei 98,4% im Training und bei 89,5% beim Test (CNN mit LSTM).

[14] 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 [53] 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-Präsidentschaftswahl 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.

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 [53]

[53] 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 wird in der Arbeit [93] ein weiteres hybrides Modell aus BERT und CNN entwickelt. Dieses trägt den Namen *MCred*.

Es wurde auf vier verschiedenen Datensätzen trainiert:

- **WELFake:** Enthält 37.106 gefälschte und 35.028 echte Nachrichten und dient als Hauptdatensatz für das MCred-Modell.
- **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-Präsidentschaftswahl 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 [28] 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 [93]

2.5.5 BERT und LSTM

[75] 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.

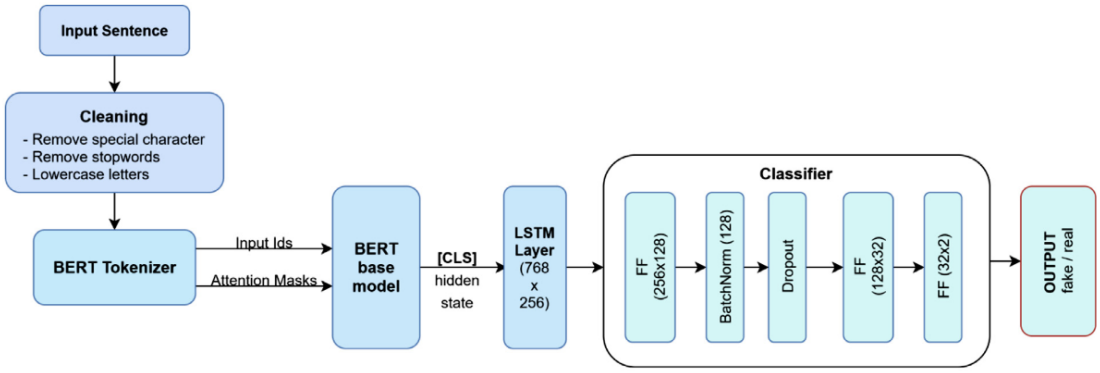

Abbildung 2.21: Architekture des vorgeschlagenen hybriden Modells [75]

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 [75]

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

[95] 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 A.1).

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). Mit etwa 8.500 Einträge ist der COVID-19-Datensatz zu klein, um das BERT-Modell effektiv und ohne Overfitting neu zu trainieren. Bei einem weiteren Training mit diesem Datensatz besteht die Gefahr, dass die vortrainierten Sprachmuster überschrieben werden. Die allgemeine Sprachkompetenz von BERT bleibt erhalten, während das darauf aufbauende BiLSTM lernt, diese Informationen für die Fake-News-Erkennung zu nutzen.

Verglichen wurde von [95] 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 2.23: Ergebnisse der verschiedenen Modelle [95]

Wie in Abbildung 2.23 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

[34] 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 2.24). 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:

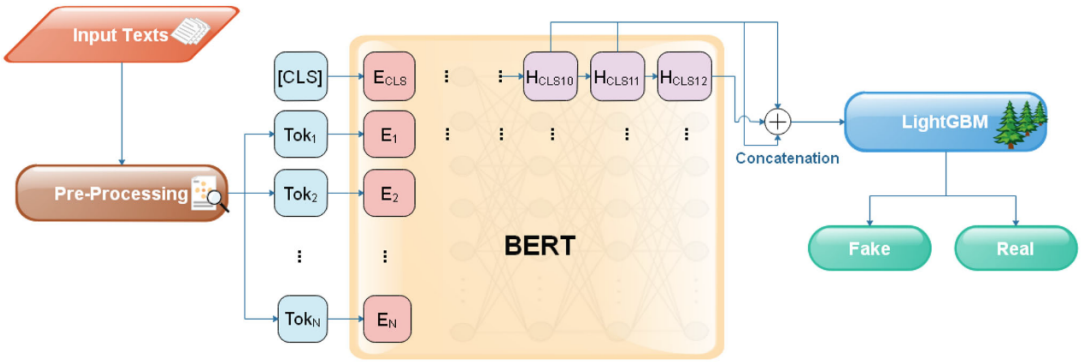

Abbildung 2.24: Architektur des hybriden Modells [34]

- **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	*F*1	Pre	Rec	Acc	*F*1	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 2.25: Ergebnisse der verschiedenen Modelle auf dem FNC-Datensatz [34]

In Abbildung 2.25 zu sehen ist, dass dieses Modell anderen gerade bei großen Datenmengen überlegen ist. [28] 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 [91] 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. Durch RoBERTAs fortgeschritteneres dynamisches Maskieren wird jedes mal genau dann ein Maskierungspattern generiert, wenn eine Sequenz einem Modell hinzugefügt wird [61].

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.

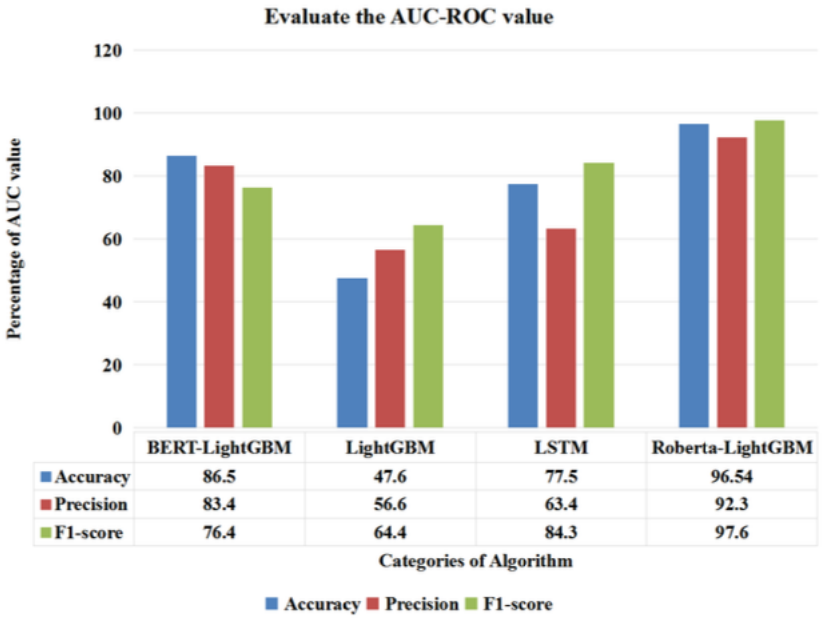

Abbildung 2.26: Vergleich der Ergebnisse der verschiedenen Modelle [91]

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 2.26).

3 Relevante Datensätze und Auswahlkriterien

Damit die im Rahmen dieser Arbeit entwickelte Anwendung in der Lage ist, Fake News zuverlässig zu erkennen, ist eine geeignete Datenbasis erforderlich. Zu diesem Zweck werden verschiedene relevante Datensätze vorgestellt, hinsichtlich ihrer Eigenschaften analysiert und im Hinblick auf ihre Eignung für das Training des Modells bewertet.

3.1 Nutzung englischer Datensätze

Eine in [85] 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 [46]. 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") wirkt sprachlich korrekt, verliert in Teilen 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 [58].

3.2 Nutzung deutscher Datensätze

In den Tabellen 3.1 und 3.2 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.

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; veraltet	Weniger geeignet
German-Fake NC	Fraunhofer-Institut für Sichere Informationstechnologie SIT	489	100 %	Enthält nur Referenzen, keine Texte → keine direkte Textauswertung möglich; veraltet	Nicht geeignet
FANG-COVID	Association for Computational Linguistics	41.242	31,97 %	Ausgewogene Klassen, vollständige Texte, viele Metadaten; veraltet	**Sehr geeignet**
DeFaktS	FZI Forschungszentrum Informatik	–	–	Kein Zugang → Nutzung ausgeschlossen	Nicht verfügbar

Tabelle 3.1: Vergleich deutscher Fake-News-Datensätze hinsichtlich Umfang, Quellenlage und praktischer Eignung

Da der DeFaktS Datensatz nicht öffentlich verfügbar ist, fällt dieser aus der Auswahl. Neben genereller Verfügbarkeit ist außerdem die Anzahl der verfügbaren Daten für das Training eines Modells relevant. Die in Kapitel 2.5.7 vorgestellten Datensätze umfassen von 20.000 bis hin zu Millionen Artikeln. Aufrund der nicht ausreichenden Größe entfällt der GermanFakeNC Datensatz.

Ein Problem des Fake News Dataset German Datensatzes (FNDG) ist die stark unausgewogene Klassenverteilung. Nur 7,24% der 64.868 Einträge sind als Fake-News ge-

Datensatz	Kontext	Relevante Features
Fake News Dataset German	nicht spezifisch	'url', 'Titel', 'Body', 'Kategorie', 'Datum', 'Quelle', 'Fake', 'Art'
German-Fake NC	nicht spezifisch	'Date', 'URL', 'False Statement 1 Location', 'False Statement 1 Index', 'Ratio of Fake Statements', 'Overall Rating'
FANG-COVID	COVID-19 Pandemie	'article', 'date', 'header', 'label', 'url'
DeFaktS	–	–

Tabelle 3.2: Vergleich deutscher Fake-News-Datensätze hinsichtlich Kontext und relevanter Features

kennzeichnet. Dadurch kann es beim Trainieren eines Modells dazu kommen, dass es überwiegend die häufiger vorkommenen 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 würde genau die Einträge verfehlen, 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 [63] 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 Klassifierung gewertet werden, auch wenn sie in neutralem oder echtem Kontext auftreten [16, 67]. 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 ein-

geschränkt 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:

1. Nur FANG-COVID mit allen 41.242 Einträgen, aber 31,97% Klassenausgewogenheit
2. FANG-COVID mit ca. 26.000 Einträgen aber dafür 50/50 Klassenausgewogenheit
3. FANG-COVID und FNDG kombiniert für bessere Generalisierung (105.110 Einträge, aber 16,94% Klassenausgewogenheit)
4. FANG-COVID und alle Fake Artikel von FNDG kombiniert für bessere Klassenausgewogenheit und mehr Daten (45.866 Einträge mit 38.82% Klassenausgewogenheit)
5. Alle Fake Einträge aus FANG-COVID und FNDG kombiniert (17.809 Fake Einträge) kombiniert mit Stichproben aus sowohl FANG-COVID als auch FNDG um 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).

Auf Grundlage dieser Analyse wird die fünfte Variante als finaler Trainingsdatensatz gewählt. Er bietet eine gute Balance aus Datenmenge, thematischer Vielfalt und Klassenausgewogenheit. Durch die Kombination beider Quellen wird die Generalisierungsfähigkeit verbessert, ohne dass das Modell durch einseitige Inhalte oder unausgewogene Klassenverteilungen verzerrt wird. Diese Auswahl eignet sich besonders für eine robuste und themenübergreifende Erkennung von Fake News.

4 Konzeption der Softwarelösung

Die entwickelte Anwendung ist aufgeteilt in zwei Komponenten. Die erste Komponente, das Machine Learning Modell, klassifiziert die Nachrichtenartikel und gibt das entsprechende Label zurück. Die zweite Komponente ist der Webagent, welcher die Artikel auf den Nachrichtenportalen liest, an das Machine Learning Modell schickt und die Antwort in das Portal integriert.

Im folgenden Kapitel wird das jeweilige Konzept dieser beiden Komponenten vorgestellt.

4.1 Konzeption des Machine Learning Modells

Zur zuverlässigen Erkennung von Fake News stellt das Machine Learning Modell die zentrale Komponente der Softwarelösung dar. In diesem Abschnitt wird zunächst das konzeptionelle Vorgehen bei der Entwicklung des Modells beschrieben. Dabei wird erläutert, welche Modellarchitekturen zum Einsatz kommen und auf welchen Grundlagen diese Auswahl basiert. Darauf aufbauend werden die wesentlichen Verarbeitungsschritte dargestellt. Diese umfassen die Datenvorverarbeitung, das Fine-Tuning der vortrainierten Transformer-Modelle, die Erzeugung semantischer Embeddings sowie deren Weiterverarbeitung mittels des LightGBM-Klassifikators. Abschließend wird die resultierende Gesamtarchitektur des Klassifikationssystems vorgestellt.

4.1.1 Auswahl und Begründung der genutzten Modelle

Um ein Modell zu entwickeln, welches möglichst gute Metriken erzielt, müssen verschiedenen Modelkombinationen getesten werden. Basierend auf den Arbeiten von [34] und [91], 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 guten Generalisierungen, niedrigem Overfitting und schnellem 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.

[87] testet verschiedene Truncation-Methoden und zeigt für BERT, dass bei Filmrezensionen und Nachrichtenartikeln eine Zusammensetzung der ersten 256 und letzten 256 Token am effektivstem 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 Layer, wobei jede Schicht die Werte weiter verfeinert und anreichert.

Um die Embeddings besser und stabiler zu repräsentieren, werden die Ausgaben der verschiedenen Hidden Layer zusammengefasst. Hierfür kann zum Beispiel der `[CLS]`-Token aus einer oder mehreren Schichten extrahiert oder ein Durchschnitt aller Token-Embeddings gebildet werden.

[87] zeigt für BERT, dass die letzten Layer die meisten Informationen beinhalten. In dieser Arbeit wird aus den letzten vier Layern ein Durchschnittswert gebildet.

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 Konzeptionelle Systemarchitektur

Die zuvor beschriebenen Verarbeitungsschritte werden in einer Gesamtarchitektur gebündelt und als Webserver bereitgestellt. Dieser Webserver übernimmt die Aufgabe, eingehende Nachrichtenartikel automatisiert zu analysieren und auf ihren Wahrheitsgehalt zu überprüfen.

Abbildung 4.1 zeigt den beispielhaften Ablauf einer solchen Klassifikation. Ein Artikel wird zunächst in einzelne Tokens zerlegt und anschließend durch ein vortrainiertes und feinjustiertes Transformer-Modell in einen semantischen Vektor überführt. Der hierbei erzeugte `[CLS]`-Vektor repräsentiert den gesamten Artikel in einem dichten, 768-dimensionalen Raum und dient als Input für das LightGBM-Modell. Dieses nimmt auf Grundlage dieses Vektors die finale Klassifikation vor. Die Vorhersage (z.B. „Artikel ist wahr“) wird anschließend dem aufrufenden System zurückgegeben.

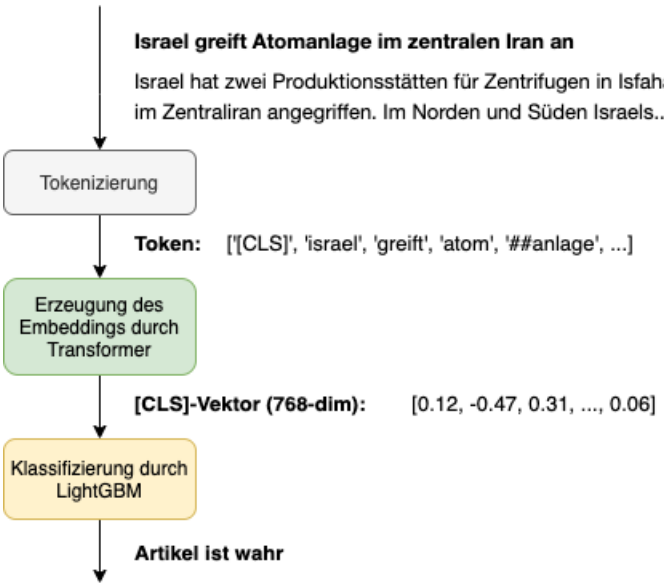

Abbildung 4.1: Beispielhafter Ablauf einer Klassifizierung eines Artikels

Diese Architektur erlaubt es, neue Texte schnell und effizient zu bewerten, ohne dass jedes Mal das komplexe Transformer-Modell zur Klassifikation herangezogen werden muss.

Stattdessen erfolgt dies über das deutlich ressourcenschonendere LightGBM-Modell. Damit ist die Lösung sowohl skalierbar als auch in Echtzeit einsetzbar.

4.2 Konzeption des Webagenten

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.

Als Beispiel die Seite Bild.de: Je nach Fenstergröße hat die Seite entweder die Domäne *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>
            <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.

5 Umsetzung der Softwarelösung

Basierend auf den vorgestellten Konzepten folgt die technische Erläuterung der Anwendung.

5.1 Implementierung des Machine Learning Modells

Die praktische Umsetzung des Machine Learning Modells basiert auf den konzeptionellen Grundlagen aus Kapitel 4. Ziel der Implementierung ist es, eine skalierbare und wiederverwendbare Pipeline zur automatisierten Klassifikation von Nachrichtenartikeln zu entwickeln.

Dazu werden zunächst mehrere vortrainierte Transformer Modelle feinjustiert (Fine-tuning). Dieser Prozess ermöglicht es, die Modelle auf die konkrete Zielaufgabe, also auf das Unterscheiden zwischen echten und gefälschten Nachrichten, zu spezialisieren. Anschließend werden aus den trainierten Modellen Embeddings extrahiert, die den semantischen Gehalt eines Artikels komprimiert abbilden. Diese Embeddings dienen als Eingabe für ein LightGBM Modell, das für die finale Klassifikation verwendet wird.

Im weiteren Verlauf dieses Abschnitts werden die technischen Details zur Modellanpassung, Generierung der Embeddings sowie zur Integration der Komponenten in eine produktionsreife Architektur beschrieben.

5.1.1 Fine-tuning der Transfomer Modelle

Für das Fine-tuning der verschiedenen Modelle werden die *Transformers*-Bibliothek von Hugging Face, die Programmiersprache 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
- XML-RoBERTa base
- XML-RoBERTa large

In A.5 angehängt, findet sich ein tabellarischer Vergleich der verschiedenen BERT- und RoBERTa-Modelle mit deren jeweiligen 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.6).

Zur dynamischen Anpassung der Lernrate während des Trainings, wird ein linearer Learning Rate Scheduler verwendet. Zu Beginn des Trainings wird die Lernrate über 10% der Trainingsschritte schrittweise erhöht, um das Modell zu stabilisieren (Warmup). 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 einzelne Repositories 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 Tranformer 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 des `attention_mask`-Vektors werden `[PAD]`-Tokens ausgeschlossen. Dieser Vektor stammt aus der *Transformers*-Bibliothek und markiert binär alle Vorkommen der `[PAD]`-Tokens. Die verbleibenden Token-Embeddings werden über die Sequenzlänge gemittelt, sodass ein einziger Vektor pro Eingabesequenz entsteht.

7. **Speicherung der Embeddings:** Die resultierenden Embeddings sowie die zugehörigen Labels werden gesammelt und in *NumPy*-Arrays konvertiert.

8. **Finaler Output:** Nach Verarbeitung aller Batches werden die 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.7). Zur Wahl der geeignetsten 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.

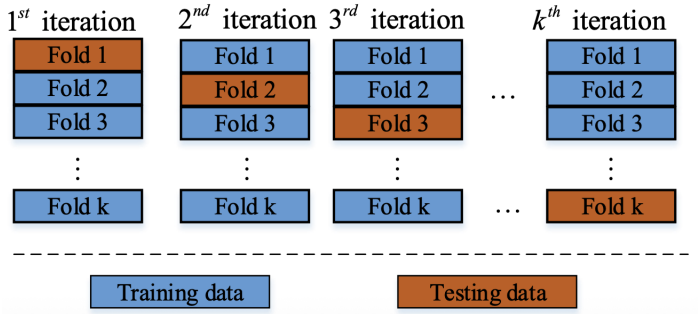

Abbildung 5.1: K-Fold-Cross-Validation [69]

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. Dadurch wird Overfitting verhindert und folglich eine bessere Generalisierung auf unbekannte Texte ermöglicht.

5.1.4 Technische Systemarchitektur

Der Webserver wird mit der Python *Starlette*-Bibliothek implementiert und mit Docker containerisiert. Nach dem Starten werden zuerst die Transformer und LightGBM Modelle geladen, danach wird ein *ready*-Flag gesetzt und der Server kann Anfragen empfangen. Die einzige Klasse ist das Singleton *EmbeddingService* (siehe Abbildung 5.2). Dieses erzeugt das Embedding und klassifiziert es anschließend. Zurückgegeben wird ein

json-Objekt mit den Informationen *label* und *probability*. Das *label* enthält den klassifizierten Binärwert (1 bedeutet Fake, 0 bedeutet Echt). Die *probability* gibt an, mit welcher Wahrscheinlichkeit der Artikel ein Fake ist (in Prozent).

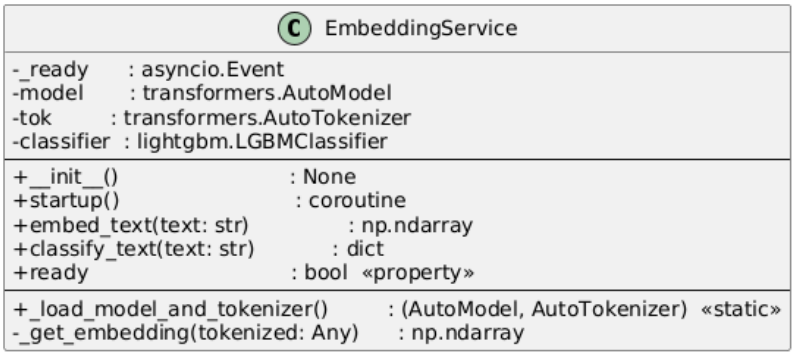

Abbildung 5.2: UML des Webservers

Eine Modellierung der Architektur des Webservers findet sich im Anhang A.3.

5.2 Implementierung des Webagenten

Zur Bestimmung des geeignetsten Tools für den Webagenten, wurden verschiedene Technologien verglichen (siehe Tabelle A.4). 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 Usercripts 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 Browseraktionen 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. Darauf basierend wird zur Implementierung des Webagents eine Chrome Extension (Manifest V3) verwendet.

Genutzt wird ein *Service Worker* ein *Content Script* pro Nachrichtenportal und ein *Popup*.

Service Worker kontrollieren eine Seite genau dann, wenn sie in der Lage sind, Netzwerkanfragen dieser Seite innerhalb ihres definierten Scopes abzufangen. Innerhalb dieses Bereichs können Service Worker dann bestimmte Aufgaben für die Seite übernehmen.

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

Nach Abschluss der Aktivierung steuern Service Worker die Seite standardmäßig erst bei der nächsten Navigation oder Seitenaktualisierung [38].

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 [36].

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 das Aufrufen von chrome.action.openPopup() ausgelöst. Pop-ups werden automatisch geschlossen, wenn der Nutzer sich auf einen Bereich des Browsers außerhalb des Pop-ups konzentriert [37].

In Abbildung A.2 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 wird 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

6.1 Leistungs-Analyse der Transformer- und LightGBM-Modelle

Tabelle 6.1 vergleicht die Accuracy- und F1-Werte der Trainings- und Testphase für alle fünf untersuchten Transformer-Modelle. Die zusätzlich angegebenen Differenzwerte (Δ) zeigen den Unterschied zwischen Test- und Trainingsergebnissen. Da die Abweichungen in beiden Metriken bei allen Modellen sehr klein ist, lässt sich daraus schließen, dass kein Overfitting vorliegt. Die Modelle generalisieren gut und zeigen eine stabile Leistung auf bislang ungesehenen Daten.

Modell	Training		Test		Δ (Test - Train)	
	Accuracy	F1	Accuracy	F1	Δ Acc	Δ F1
XLM-RoBERTa-Large	0.9780	0.9780	0.9795	0.9795	+0.0015	+0.0015
XLM-RoBERTa-Base	0.9730	0.9729	0.9717	0.9716	−0.0013	−0.0013
RoBERTa-Large	0.9795	0.9795	0.9765	0.9764	−0.0030	−0.0031
RoBERTa-Base	0.9771	0.9771	0.9751	0.9751	−0.0020	−0.0020
BERT-Base-Uncased	0.9507	0.9505	0.9533	0.9531	+0.0026	+0.0026

Tabelle 6.1: Vergleich von Training und Test: Accuracy und F1 zur Überprüfung von Overfitting

Tabelle 6.2 zeigt die Testergebnisse der fünf untersuchten Transformer Modelle nach dem Fine-Tuning auf dem Klassifikationsdatensatz. Dargestellt sind Accuracy, F1-Score, Loss sowie die Evaluationszeit des Testdatensatzes. Das Modell XLM-RoBERTa-Large erzielt mit einer Accuracy und einem F1-Score von jeweils 97,95% die besten Resultate, gefolgt von RoBERTa-Large und RoBERTa-Base. Das schwächste Ergebnis liefert BERT-Base-Uncased mit einem Accuracy-Wert von 95,33%. Insgesamt zeigt sich, dass die *Large*-Varianten der Tranformer Modelle bessere Ergebnisse erzielen, allerdings auch mit einem erhöhten Evaluationsaufwand verbunden sind.

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 6.2: Testergebnisse der Transformer Modelle nach dem Fine-Tuning

In Tabelle 6.3 dargestellt, sind die Metriken verschiedener LightGBM Modelle, welche auf den neu erzeugten Embeddings der verschiedenen Transformer-Modellen trainiert wurden. Dabei wurden jeweils die letzten vier Schichten der Modelle gemittelt und als Input für ein untrainiertes LightGBM Modell verwendet. Auch in diesem Setup erzielt XLM-RoBERTa-Large mit einer Accuracy von 97,84% und einem F1-Score von 97,73% die besten Ergebnisse. Es folgen die *Base*-Varianten von XLM-RoBERTa und RoBERTa mit sehr ähnlichen Werten. Deutlich schwächer schneidet BERT-Base-Uncased ab. Insgesamt zeigt sich, dass die Qualität der Repräsentationen der Transformer-Modelle mit komplexer Architektur und zunehmender Vokabulargröße der jeweiligen Transformer steigt. Die Qualität hat außerdem einen wesentlichen Einfluss auf die nachgelagerte Klassifikationsleistung hat.

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 6.3: Testergebnisse der LightGBM-Modelle nach dem Training auf den Embeddings

Tabelle 6.4 vergleicht die Accuracy- und F1-Werte der Transformer und LightGBM Modelle. Der Differenzwert (Δ) zeigt den Unterschied zwischen den beiden Modellen. Es zeigt sich, dass die direkt feinjustierten Transformer-Modelle in fast allen Fällen leicht bessere Ergebnisse erzielen als die Kombination aus Embeddings und LightGBM.

Ein möglicher Grund für die leicht schlechteren Metriken bei der Verwendung von LightGBM könnte in der hohen semantischen Dichte und Dimension der Transformer-Embeddings liegen. Diese enthalten sehr viele komplexe, kontextuelle Informationen, die zwar für

Modell	Transformer		LightGBM		Δ (LGBM - TF)	
	Accuracy	**F1**	**Accuracy**	**F1**	Δ Acc	Δ F1
XLM-RoBERTa-Large	0.9795	0.9795	0.9784	0.9773	−0.0011	−0.0022
RoBERTa-Large	0.9765	0.9764	0.9753	0.9738	−0.0012	−0.0026
RoBERTa-Base	0.9751	0.9751	0.9729	0.9713	−0.0022	−0.0038
XLM-RoBERTa-Base	0.9717	0.9716	0.9753	0.9738	+0.0036	+0.0022
BERT-Base-Uncased	0.9533	0.9531	0.9472	0.9439	−0.0061	−0.0092

Tabelle 6.4: Vergleich von Transformer- und LightGBM-Modellen: Accuracy, F1 und Differenz

überwachte Modelle direkt nutzbar sind, klassische Modelle wie LightGBM jedoch nur eingeschränkt verarbeiten können. [77] erklärt, dass die Architektur von BERT für die Anwendung in weiteren unüberwachte Verfahren problematisch ist. Es lässt sich vermuten, dass Gleiches auch für überwachte Modelle gilt, sofern diese nicht speziell auf die Strukturen der Transformer-Embeddings ausgelegt sind.

6.2 Vergleich mit verwandten Arbeiten

Tabelle 6.5 vergleicht die Accuracy- und F1-Scores der in dieser Arbeit entwickelten Modelle mit den besten Resultaten zweier verwandter Studien. Während [34] ihre Modelle auf drei unterschiedlichen Fake-News-Datensätzen getestet hat (ISOT, TI-CNN, FNC), stammen die Ergebnisse aus [91] vom FNDD-Datensatz. In dieser Arbeit wurde ein weiterer selbst entwickelter Datensatz verwendet. Aufgrund dieser Unterschiede ist ein direkter Vergleich nur bedingt sinnvoll.

Auffällig ist, dass in den Arbeiten von [34] und [91] ausschließlich Ergebnisse für klassifikatorgestützte Ansätze (z.,B. BERT-Embeddings mit LightGBM oder LSTM) gezeigt werden. Es fehlen vergleichbare Kennzahlen für direkt feinjustierte Transformer-Modelle wie sie in dieser Arbeit durchgeführt wurden. Ein direkter Leistungsvergleich mit reinem Transformer-Fine-Tuning ist folglich nicht möglich.

Trotzdem lässt sich beobachten, dass das in dieser Arbeit durchgeführte Fine-Tuning des XLM-RoBERTa-Large-Modells mit einer Accuracy und einem F1-Score von jeweils 97,95% sehr gute Ergebnisse erzielt und damit nahe an den besten LightGBM Kombinationen der Vergleichsarbeiten liegt. Auch das LightGBM-Modell auf Basis der XLM-

RoBERTa-Embeddings erreicht mit 97,84% (Accuracy) und 97,73% (F1) ein konkurrenzfähiges Niveau.

Die teils noch höheren Werte in [34] könnten unter anderem auf Unterschiede in den Datensätzen zurückzuführen sein. So ist beispielsweise der ISOT-Datensatz stark strukturiert und enthält viele sich wiederholende Phrasen, was Klassifizierungen durch Embedding-Modelle begünstigen kann [4].

Arbeit	**Datensatz**	**Bestes Modell**	**Acc. (%)**	**F1 (%)**
[34]	ISOT	BERT + LightGBM	99.88	99.88
[34]	FNC	BERT + LightGBM	99.06	99.05
Diese Arbeit	Eigener Datensatz	XLM-RoBERTa-Large	97.95	97.95
Diese Arbeit	Eigener Datensatz	XLM-RoBERTa-Large + LightGBM	97.84	97.73
Diese Arbeit	Eigener Datensatz	RoBERTa-Large + LightGBM	97.53	97.38
[34]	TI-CNN	BERT + LightGBM	96.94	97.42
[91]	FNDD	RoBERTa + LightGBM	96.54	97.60
Diese Arbeit	Eigener Datensatz	BERT + LightGBM	94.72	94.39

Tabelle 6.5: Vergleich der erzielten Accuracy- und F1-Scores mit verwandten Arbeiten

Zusammenfassend zeigen die Ergebnisse, dass die in dieser Arbeit verfolgten Methoden sowohl im direkten Fine-Tuning als auch im kombinierten LightGBM-Ansatz mit aktuellen Forschungsarbeiten konkurrenzfähig sind. Zudem wird aber auch deutlich, dass ein sinnvoller Vergleich verschiedener Modellarchitekturen sehr schwer fällt. So liefert zum Beispiel die Kombination von BERT und LightGBM in dieser Arbeit mit 94,72% (Accuracy) das schlechteste Ergebniss, während die gleiche Kombination in [34] 5,16% mehr erzielt. Mögliche Ursachen hierfür können neben den verwendeten Datenbasen die unterschiedlichen Konfigurationen der Hyperparameter beim Modelltraining oder das Verfahren der Embedding-Bildung sein.

6.3 Ergebnisse der finalen Anwendung

Die bereitgestellte Anwendung liefert einen automatischen Klassifizierungsservice für Nachrichtenartikel auf den Seiten *BILD.de*, *taz.de* und *spiegel.de*.

Die in Abbildung 6.1 dargestellten Beispiele verdeutlichen, dass die vorgenommenen Klassifizierungen noch Optimierungspotenzial aufweisen. Ein möglicher Grund hierfür liegt in der unzureichenden Abdeckung aktueller Themen in den für das Training verwendeten Datensätzen. Infolgedessen verfügen die Modelle nicht über ausreichenden Kontext, um die abgefragten Domänen adäquat bewerten zu können.

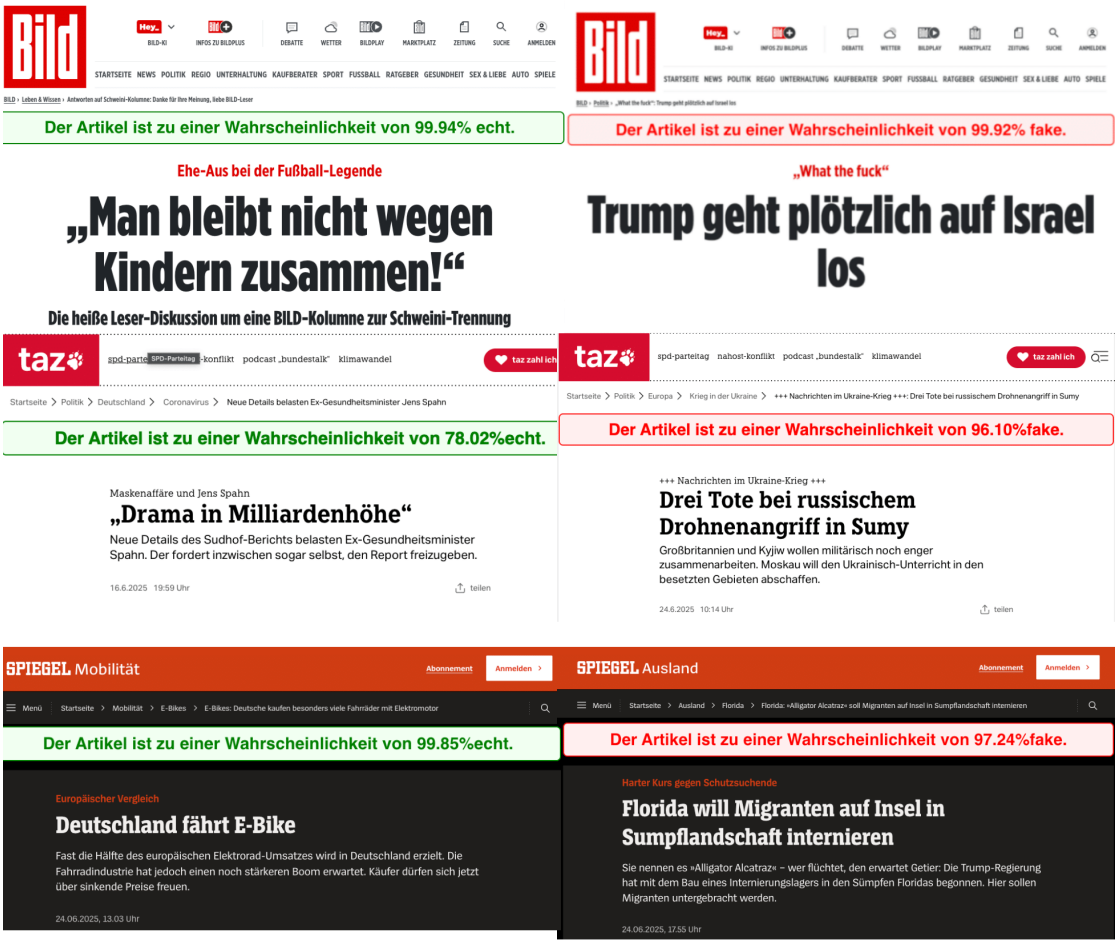

Abbildung 6.1: Screenshots der Anwendung auf den verschiedenen Nachrichtenportalen - faktisch sind alle Artikel korrekt

Ein exemplarischer Fall ist der Artikel auf BILD.de mit dem Titel „Trump geht plötzlich auf Israel los“. Obwohl der Inhalt des Artikels faktisch korrekt ist, wird er vom Modell fälschlicherweise als falsch eingestuft.

Diese Fehleinschätzung könnte durch semantische oder stilistische Merkmale in Überschrift und Artikel bedingt sein. Um diesen Einfluss näher zu untersuchen, wäre es denkbar, Titelvariationen wie „Trump kritisiert Israel - eine Kehrtwende in seiner bisherigen Haltung“ mit entsprechend angepasstem Artikelinhalt zu testen.

7 Fazit

Ziel dieser Arbeit war die Konzeption und prototypische Umsetzung einer Anwendung zur automatisierten Erkennung von Falschmeldungen in Online-Nachrichtenartikeln. Aufbauend auf aktuellen gesellschaftlichen Herausforderungen im Umgang mit Fake News wurde ein Prototyp konzipiert, implementiert und evaluiert, der moderne Methoden des Natural Language Processing (NLP) mit klassischem maschinellen Lernen kombiniert.

Im Mittelpunkt der Arbeit standen zwei Modellierungsansätze: Einerseits wurden vortrainierte Transformer-Modelle (BERT, RoBERTa, XLM-RoBERTa) direkt per Fine-Tuning für die binäre Klassifikation optimiert. Andererseits wurden Embeddings dieser Modelle extrahiert und als Feature-Vektor für LightGBM-Modelle verwendet. Ergänzend dazu wurde ein eigener deutschsprachiger Datensatz zusammengestellt, um die Modelle praxisnah zu trainieren und zu testen. Zur Umsetzung kamen aktuelle Technologien wie Python, Hugging Face Transformers sowie PyTorch zum Einsatz.

Die Evaluation zeigte, dass insbesondere das Modell XLM-RoBERTa-Large im direkten Fine-Tuning mit einer Accuracy und einem F1-Score von jeweils 97,95% die besten Resultate erzielte. Auch im hybriden Ansatz mit LightGBM auf Basis von Transformer-Embeddings erreichte dieses Modell hohe Werte (Accuracy: 97,84%). Über alle Experimente hinweg zeigte sich, dass die *Large*-Varianten der Modelle tendenziell bessere Resultate liefern als die *Base*-Versionen. Allerdings mit erhöhtem Rechenaufwand.

Entgegen der ursprünglichen Annahme, dass die Kombination aus Transformer Embeddings und LightGBM Modellen eine bessere Generalisierungsleistung ermöglichen könnte, schnitten die hybriden Modelle insgesamt leicht schlechter ab als die direkt feinjustierten Transformer. Diese Beobachtung legt nahe, dass das direkte Fine-Tuning, trotz potenziell höherer Rechenkosten, die leistungsstärkere Strategie für die Fake-News-Klassifikation darstellt. Der Vergleich mit verwandten Studien unterstreicht die Wettbewerbsfähigkeit der entwickelten Modelle, auch wenn unterschiedliche Datensätze und Modellkonfigurationen einen direkten Leistungsvergleich erschweren.

Mit dem entwickelten funktionsfähigen Prototypen wurde ein wichtiges Ziel der Arbeit erreicht: Eine praxistaugliche Anwendung, die durch automatisierte Klassifikation Fake News erkennen kann.

Dennoch wurde im Verlauf der Arbeit deutlich, dass insbesondere im Kontext der Fake-News-Erkennung ein breit gefächerter und aktueller Trainingsdatensatz essenziell ist. Nur durch eine thematisch vielfältige und zeitnahe Datenbasis kann das Modell ausreichend kontextuelles Wissen aufbauen, um Nachrichteninhalte zuverlässig und differenziert klassifizieren zu können.

Insgesamt beweist diese Arbeit, dass transformerbasierte Modelle alleinstehend oder in Kombination mit modernen Frameworks eine effektive Grundlage für die automatisierte Fake-News-Erkennung sein können.

8 Ausblick

Die in dieser Arbeit entwickelte Anwendung bildet eine solide Grundlage für die automatisierte Klassifikation von Nachrichtenartikeln. Gleichzeitig ergeben sich aus den gewonnenen Erkenntnissen viele Ansatzpunkte für eine weiterführende Forschung und technische Weiterentwicklung.

Ein zentraler Aspekt für zukünftige Arbeiten ist der Einsatz eines aktuelleren, domänenübergreifenden Datensatzes, der verschiedene Themengebiete abdeckt und damit eine verbesserte Generalisierungsfähigkeit der Modelle ermöglicht. Der in dieser Arbeit verwendete, eigens zusammengestellte Datensatz beinhaltet einen starken Fokus auf den COVID-19 Pandemie Kontext und ist außerdem stark veraltet. Eine Ausweitung auf weitere thematische und aktuellere Domänen könnte die Robustheit der Modelle signifikant erhöhen.

Des Weiteren bietet sich der Einsatz leistungsfähigerer Transformer Modelle wie XLM-RoBERTa-XL an. Dieses Modell stellt eine noch größere Variante des in dieser Arbeit bereits erfolgreich getesteten XLM-RoBERTa-Large dar und könnte insbesondere bei umfangreicheren Texten eine noch tiefere semantische Repräsentation ermöglichen. Gescheitert ist die Verwendung dieses Modells an nicht ausreichenden Hardwarekapazitäten.

Ein bedeutender Limitationsfaktor vieler aktueller Transformer Modelle ist die Beschränkung auf eine maximale Inputlänge von 512 Tokens. Künftig könnten entweder Modelle mit erweiterter Kontextlänge (z.B. Longformer oder BigBird) oder Strategien zur Input-Segmentierung, wie die Kombination von Head- und Tail-Abschnitten eines Dokuments [87], eingesetzt werden. Auf diese Weise ließen sich längere Nachrichtenartikel ohne Informationsverlust verarbeiten.

Auch auf der Ebene der Repräsentationserzeugung aus den Transformer Modellen bieten sich Optimierungsmöglichkeiten. In dieser Arbeit wurden die Mittelwerte der letzten vier Hidden Layers genutzt, doch es existieren weitere Alternativen wie z.B. die Verwendung des `[CLS]`-Tokens, der maximale Layer-Wert, oder unterschiedliche Gewichtungen der

Layer-Repräsentationen. Ein systematischer Vergleich dieser Methoden könnte zu einer verbesserten Repräsentationsqualität führen.

Darüber hinaus wäre ein Vergleich mit alternativen Deep-Learning Modellarchitekturen von Interesse, wie zum Beispiel mit dem Ansatz FakeBERT [53], welcher CNNs mit BERT kombiniert. Besonders vielversprechend erscheint die Kombination von CNNs mit XLM-RoBERTa als Embedding-Modell, um die semantische Tiefe transformerbasierter Repräsentationen mit den extraktiven Fähigkeiten von Convolutional Layers zu verbinden. Generell können Kombinationen aus allen in Kapitel 2 genannten Modellen erprobt werden.

Auf Anwendungsebene eröffnet sich weiteres Potenzial durch das Deployment der entwickelten Anwendung, um diese nicht nur lokal zu demonstrieren. Durch die Integration zusätzlicher Nachrichtenportale ließe sich ein vielfältiges Spektrum des deutschsprachigen Journalismus automatisiert erfassen, analysieren und klassifizieren. Ergänzend dazu könnte eine eigene Webplattform entwickelt werden, auf der Nutzer:innen Artikel posten und direkt eine Echtzeit-Klassifikation erhalten. Dies würde nicht nur die Nutzbarkeit verbessern, sondern auch einen konkreten gesellschaftlichen Mehrwert leisten.

Insgesamt bieten sich zahlreiche Perspektiven zur Weiterentwicklung sowohl im Hinblick auf Modellarchitektur als auch auf Anwendungsintegration. Die hier vorgestellte Arbeit kann somit als Ausgangspunkt für praxisnahe, hochskalierbare und transparente Systeme zur Bekämpfung von Desinformation im digitalen Raum dienen.

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] Abdullah Marish Ali, Fuad A. Ghaleb, Bander Ali Saleh Al-Rimy, Fawaz Jaber Alsolami, and Asif Irshad Khan. Deep ensemble fake news detection model using sequential deep learning technique. *Sensors*, 22(18), 2022. ISSN 1424-8220. doi: 10.3390/s22186970. URL https://www.mdpi.com/1424-8220/22/18/6970.

[5] 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/pfclh2k6tffsdeh5ezp3qtdkoy/access/wayback/http://202.52.52.28/index.php/kinetik/article/download/1066/pdf.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] Joel Barnard. Was sind worteinbettungen?, 2024. https://www.ibm.com/de-de/think/topics/word-embeddings [Accessed: 18.05.2025].

[11] Benjamin Franklin. "supplement to the boston independent chronicle," before 22 april 1782. Founders Online, National Archives, 1782. URL https://founders.archives.gov/documents/Franklin/01-37-02-0132. Original source: The Papers of Benjamin Franklin, vol. 37, March 16 through August 15, 1782, ed. Ellen R. Cohn. New Haven and London: Yale University Press, 2003, pp. 184–196.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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. ISBN 9781450342322. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145/2939672.2939785.

[18] KENNETH WARD CHURCH. Word2vec. *Natural Language Engineering*, 23(1): 155–162, 2017. doi: 10.1017/S1351324916000334.

[19] 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.

[20] 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.

[21] 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.

[22] IBM Corporation. Bag of words explained. https://www.ibm.com/think/topics/bag-of-words, 2024. Accessed: 2025-05-16.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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. *Heliyon*, 10(16), 2024. doi: 10.1016/j.heliyon.2024.e35865. URL https://doi.org/10.1016/j.heliyon.2024.e35865.

[29] 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.

[30] Sheng Dong, Afaq Khattak, Irfan Ullah, Jibiao Zhou, and Arshad Hussain. Predicting and analyzing road traffic injury severity using boosting-based ensemble learning models with shapley additive explanations. *International Journal of Environmental Research and Public Health*, 19:2925, 03 2022. doi: 10.3390/ijerph19052925.

[31] 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].

[32] 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.

[33] 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.

[34] 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.

[35] Hugging Face. Wordpiece tokenization, 2025. https://huggingface.co/learn/llm-course/chapter6/6 [Accessed: 18.05.2025].

[36] Chrome for Developers. Content scripts. https://developer.chrome.com/docs/extensions/develop/concepts/content-scripts, 2025. Zugriff am 11. Mai 2025.

[37] Chrome for Developers. Add popup. https://developer.chrome.com/docs/extensions/develop/ui/add-popup, 2025. Zugriff am 11. Mai 2025.

[38] Chrome for Developers. A service worker's life. https://developer.chrome.com/docs/workbox/service-worker-lifecycle, 2025. Zugriff am 11. Mai 2025.

[39] Bundeszentrale für politische Bildung (bpb). Fake news, 2022. URL https://www.bpb.de/kurz-knapp/lexika/das-junge-politik-lexikon/320271/fake-news/.

[40] Yan Gao. Federated xgboost with bagging aggregation, November 2023. URL https://flower.ai/blog/2023-11-29-federated-xgboost-with-bagging-aggregation/. Zugriff am 14.06.2025.

[41] Benyamin Ghojogh and Ali Ghodsi. Attention mechanism, transformers, bert, and gpt: Tutorial and survey. working paper or preprint, December 2020. URL https://hal.science/hal-04637647.

[42] 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.

[43] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2nd edition, 2009. ISBN 9780387848570.

[44] 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.

[45] Alice Herman. 'they're eating the cats': Trump rambles falsely about immigrants in debate. *The Guardian*, September 2024. URL https://www.theguardian.com/us-news/article/2024/sep/10/trump-springfield-pets-false-claims.

[46] 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.

[47] 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.

[48] 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.

[49] Jasmine Grand Huo Jingnan. Jd vance spreads debunked claims about haitian immigrants eating pets. *NPR*, September 2024. URL https://www.npr.org/2024/09/10/nx-s1-5107320/jd-vance-springfield-ohio-haitians-pets.

[50] 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-Comparati

ve-Analysis-of-Sentiment-Classification-Models-for-Improved-Performance-Optimization.pdf.

[51] Vikramaditya Jakkula. Tutorial on support vector machine (svm). *School of EECS, Washington State University*, 37(2.5):3, 2006.

[52] 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.

[53] 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.

[54] 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.

[55] KiKaBeN. Transformers encoder decoder, December 2021. URL https://kikaben.com/transformers-encoder-decoder/. Veröffentlicht am 12. Dezember 2021.

[56] Juhani Kivimäki, Jakub Białek, Wojtek Kuberski, and Jukka K. Nurminen. Performance estimation in binary classification using calibrated confidence, 2025. URL https://arxiv.org/abs/2505.05295.

[57] 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/.

[58] 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.

[59] 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.

[60] 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.

[61] 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.

[62] Josh Marcus. 'they're just not true': Springfield officials furious as trump's migrant pet eating lie causes bomb threats and school closures. *The Independent*, September 2024. URL https://www.independent.co.uk/news/world/americas/us-politics/trump-jd-vance-ohio-haitians-b2612553.html.

[63] 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/.

[64] Mike Wendling Merlyn Thomas. Trump repeats baseless claim about haitian immigrants eating pets. *BBC News*, September 2024. URL https://www.bbc.com/news/articles/c77l28myezko.

[65] 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.

[66] 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.

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] 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/.

[72] 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.

[73] 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.

[74] 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.

[75] 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.

[76] 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.

[77] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410.

[78] 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.

[79] 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.

[80] 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.

[81] 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 fortext.net, offizielle Publikation am 30.10.2024.

[82] 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.

[83] 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.

[84] 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.

[85] 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.

[86] 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.

[87] 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.

[88] 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.

[89] 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.

[90] UNESCO and Erich Brost Institute for International Journalism. *Journalismus, Fake News und Desinformation: Handbuch für Journalistenausbildung und training.* UNESCO, 2022. ISBN 978-92-3-000180-3. URL https://unesdoc.unesco.org/ark:/48223/pf0000380827.

[91] Rajkumar V and Priyadharshini 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.

[92] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

[93] 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.

[94] 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.

[95] 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.

[96] 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.

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.

Tool	**Verwendung**
LaTeX	Textsatz- und Layout-Werkzeug zur Erstellung des Dokuments
ChatGPT	KI-gestützter Assistent zur Textgenerierung, Überarbeitung und Ideenfindung
ScholarGPT	Spezialisierte KI-Version zur Analyse wissenschaftlicher Quellen
Gemini	KI-Tool zur Unterstützung bei Recherche und Programmierung
Draw.io	Tool zur Erstellung von Diagrammen und technischen Skizzen
Google Colab	Cloudbasierte Python-Umgebung für Datenanalyse und Modellierung
Visual Studio Code	Quellcode-Editor zur Entwicklung und Bearbeitung von Programmdateien
Git	Versionskontrollsystem zur Nachverfolgung von Änderungen im Quellcode

Tabelle A.1: Verwendete Hilfsmittel und Werkzeuge

In Tabelle A.2 folgen die verwendeten Bibliotheken und Frameworks.

Bibliothek / Framework	**Verwendung**
Python	Programmiersprache zur Analyse, Automatisierung und Modellierung
pandas	Datenmanipulation und -analyse in Tabellenform
matplotlib	Erstellung von Diagrammen und Visualisierungen
numpy	Unterstützung für numerische Operationen und Arrays
scikit-learn	Implementierung klassischer ML-Algorithmen (z.B. Klassifikation)
transformers	Zugriff auf vortrainierte Sprachmodelle (z.B. BERT, GPT)
torch	Deep-Learning-Framework zur Implementierung neuronaler Netze
Hugging Face	Plattform und Bibliothek zur Bereitstellung von NLP-Modellen
Manifest v3	Technische Spezifikation für die Erstellung von Chrome-Erweiterungen
HTML / CSS / JavaScript	Webtechnologien zur Umsetzung der Benutzeroberfläche

Tabelle A.2: Verwendete Bibliotheken und Frameworks

A.2 Deklaration zur Nutzung von KI-gestützten Tools

In dieser wissenschaftlichen Arbeit wurden Künstliche-Intelligenz (KI-Technologien) zur Unterstützung verschiedener Aspekte der Forschung eingesetzt. Die Nutzung umfasste unter anderem die Analyse und Auswertung von Literatur, die Unterstützung bei der Datenauswertung sowie die Generierung von Ideen und Inhalten.

Es wird ausdrücklich darauf hingewiesen, dass die endgültige Verantwortung für die inhaltliche Richtigkeit, die kritische Reflexion und die Interpretation der Ergebnisse beim Autor dieser Arbeit liegt.

Die KI diente lediglich als Werkzeug und nicht als Ersatz für das kritische und analytische Denken des Forschenden.

Tabelle A.3 zeigt eine explizite Auflistung an welchen Stellen KI-Technologien verwendet wurden.

Abschnitt / Kapitel	**Art der Unterstützung durch KI**
Kapitel 1	• Ideenfindung zum Aufbau des Kapitels • Zusammenfassen wissenschaftlicher Arbeiten, Definitionen und Nachrichtenartikeln • Formulierung von Einleitungen
Kapitel 2	• Formulierung von Einleitungen • Zusammenfassen wissenschaftlicher Arbeiten • Ausformulierung von Stichpunkten • Unterstützung bei der sprachlichen Glättung und Terminologieerklärung zu den verschiedenen Modellen • Erstellung von Tabelle 2.1 und Tabelle 2.2 • Schrittweise Erklärung der Funktionalität von LightGBM, Self-Attention und Hidden-Layer • Beispiel WordPiece Embedding und BPE • Self-Attention Beispiel in Kapitel 2.3.1 • Übersetzung der Dokumentation von RoBERTa in Kapitel 2.3.2
Kapitel 3	• RKI-Satz Beispiel in Kapitel 3.1
Kapitel 4	• Formulierung von Einleitungen und Beschreibungen
Kapitel 5	• Formulierung von Einleitungen • Schrittweise Erklärung des Codes in Kapitel 5.1.2 • Erstellung der UML Klasse (Abbildung 5.2) • Erstellung von Tabelle A.4
Kapitel 6	• Erstellung der Latex Tabellen anhand von Logging Daten und Outputs • Teilweise Erstellungen von schriftlichen Beschreibungen und Vergleichen der Tabellen
Kapitel 7 & Kapitel 8	Ausformulieren von gesammelten Stichpunkten

Tabelle A.3: Dokumentation der Nutzung KI-gestützter Hilfsmittel

A.3 Abbildungen

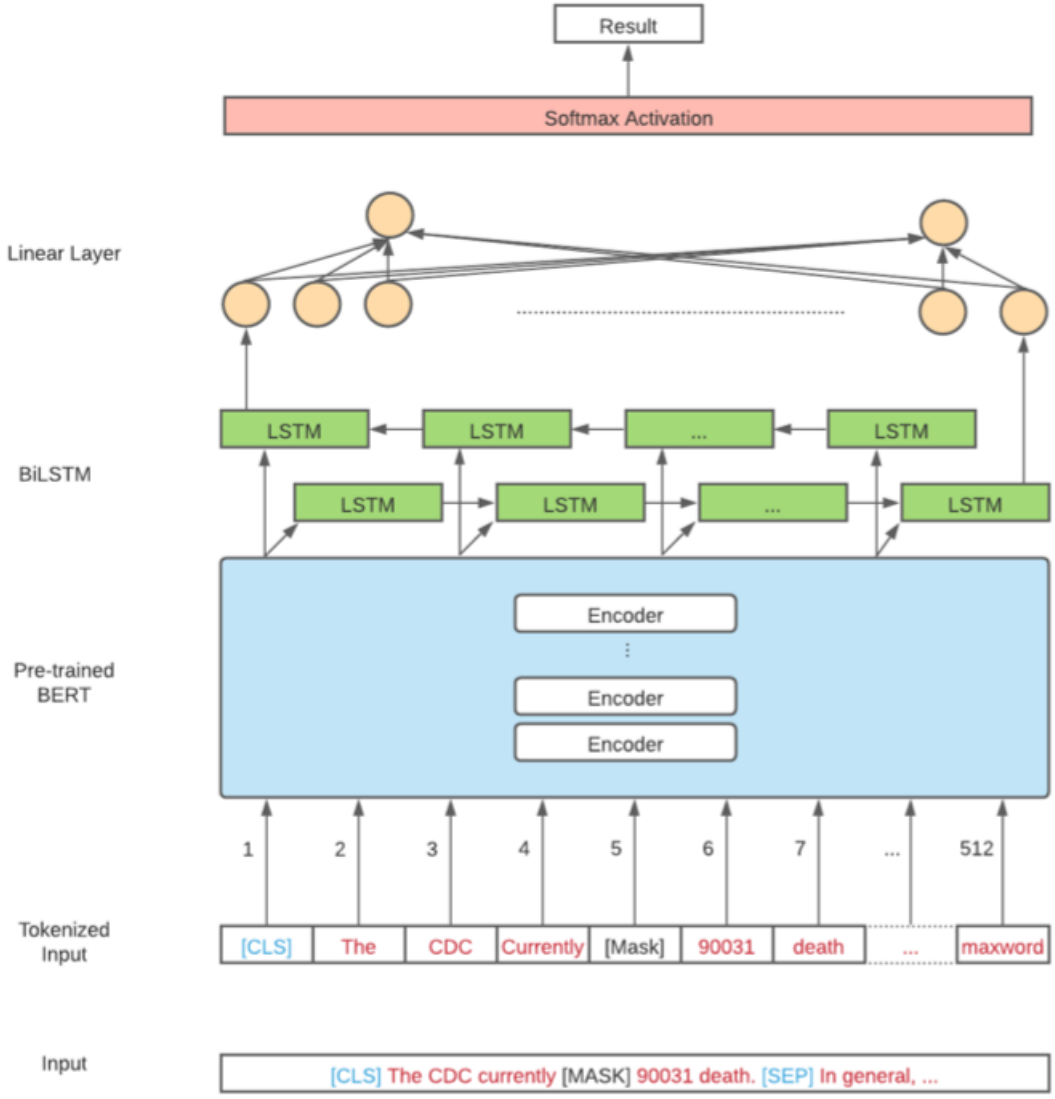

Abbildung A.1: Architektur des hybriden Modells [95]

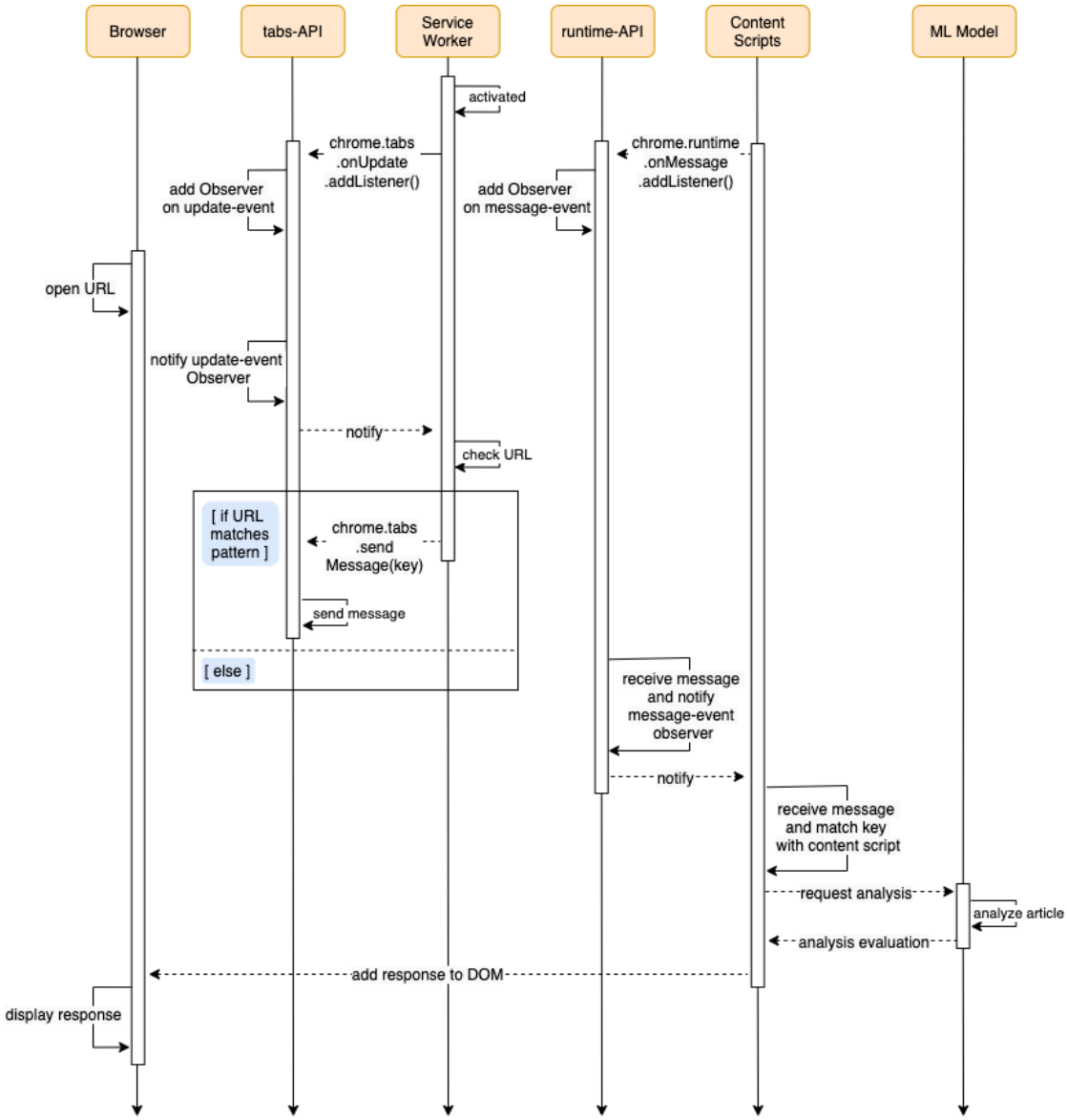

Abbildung A.2: Sequenzdiagramm Webagent

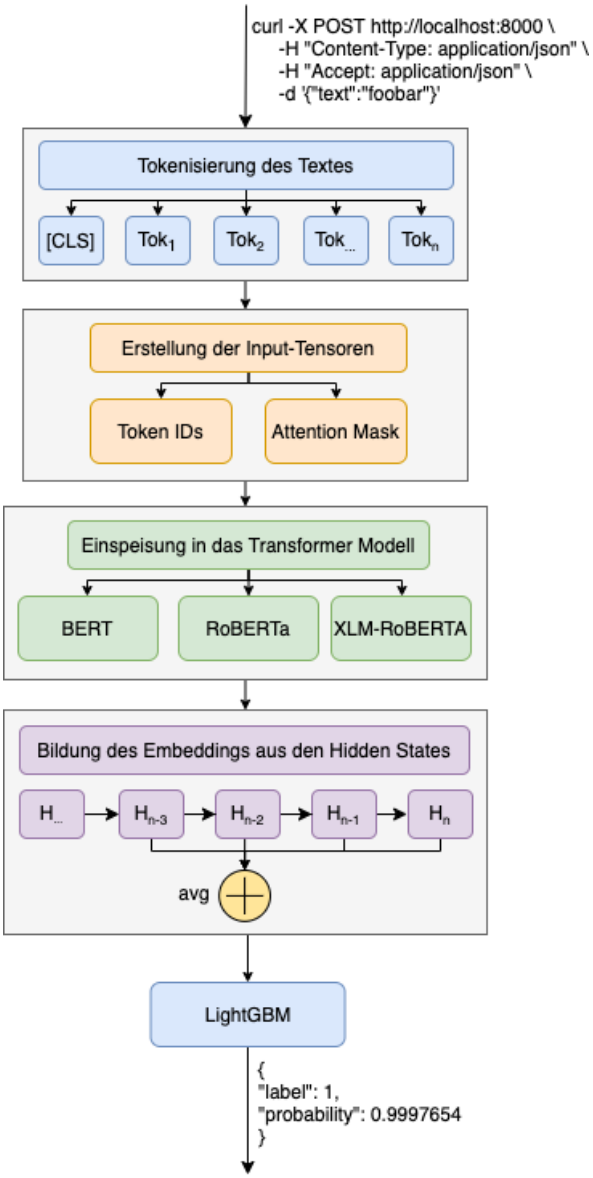

Abbildung A.3: Gesamtarchitektur des Webservers

A.4 Tabellen

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
Entwickler-kontrolle über UI	Hoch	Mittel	Hoch	Mittel

Tabelle A.4: 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	Masked LM	Masked LM	Masked LM	Multilinguales Masked LM	Multilinguales Masked LM
Sprachumfang	Englisch	Englisch	Englisch	Multilingual	Multilingual

Tabelle A.5: 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.6: Überblick über die gewählten Hyperparameter der Transformer-Modelle

Parameter	Wert	Beschreibung
Ziel (`objective`)	`binary`	Binäre Klassifikation (z. B. „echt“ oder „fake“).
Metrik (`metric`)	`binary_logloss`	Verlustfunktion zur Bewertung der Modellgüte während des Trainings.
Boosting-Typ (`boosting_type`)	`gbdt`	Verwendung von Gradient Boosted Decision Trees als Lernverfahren.
Prozessor-Kerne (`n_jobs`)	`-1`	Nutzt alle verfügbaren CPU-Kerne für paralleles Training.
Lernrate (`learning_rate`)	`0.0865`	Schrittweite beim Anpassen der Modellgewichte.
Anzahl Blätter (`num_leaves`)	`63`	Maximale Anzahl von Blättern pro Entscheidungsbaum.
Maximale Tiefe (`max_depth`)	`20`	Begrenzt die Tiefe der Bäume zur Kontrolle der Komplexität.
Min. Samples 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 zufällig ausgewählt werden.
L_1-Regularisierung (`reg_alpha`)	`0.2989`	Bestraft große Gewichtswerte zur Förderung einfacher Modelle.
L_2-Regularisierung (`reg_lambda`)	`0.4609`	Stabilisiert das Modell durch Bestrafung großer Gewichtssummen.
Zufallsstart (`random_state`)	`42`	Sichert Reproduzierbarkeit der Ergebnisse.
Anzahl Bäume (`n_estimators`)	`2000`	Maximale Anzahl an Bäumen; Early Stopping begrenzt effektiv.

Tabelle A.7: 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