Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einleitung

1.1 Motivation

Laut Duden ist eine Bewertung [Dud26] a) „das Bewerten; das Bewertetwerden“ und b) „sprachliche Äußerung, durch die etwas, jemand bewertet wird“.

Unser Leben ist heute kaum noch ohne Bewertungen vorstellbar. Sie begegnen uns überall - in Online-Shops, ebenso wie in kurzen Kommentaren, etwa wenn eine Speise als zu salzig beschrieben wird. Eine Bewertung ist dabei nichts anderes als eine persönliche Meinung oder eine subjektive Einschätzung eines Gegenstandes.

Doch warum sind Bewertungen heutzutage so wichtig? Ohne Rückmeldungen von Nutzerinnen und Nutzern wäre eine kontinuierliche Weiterentwicklung von Produkten und Dienstleistungen kaum möglich, da die Sicht der Kundschaft fehlen würde. Für Unternehmen bieten Bewertungen die Möglichkeit, ihre Produkte und Dienstleistungen zu verbessern, Kundenbeziehungen zu stärken und Vertrauen aufzubauen.

Bewertungen können sowohl positiv als auch negativ sein und enthalten oft subjektive Einschätzungen. Eine große Herausforderung besteht darin, aus den enormen Mengen an verfügbaren Textdaten genau diejenigen Informationen zu identifizieren, die für eine fundierte Beurteilung relevant sind. Aufgrund dieser Datenmengen ist eine manuelle Analyse kaum noch praktikabel, weshalb automatische Verfahren der Sentiment-Analyse eine zentrale Rolle spielen.

1.2 Ziel der Arbeit

Ziel dieser Arbeit ist es, verschiedene Ansätze der Sentiment-Analyse zur Auswertung von Kundenfeedback zu untersuchen und miteinander zu vergleichen. Dabei soll analysiert werden, welche Methoden sich für deutsch- und englischsprachige Texte eignen und wie sich diese hinsichtlich ihrer Klassifikationsleistung und ihres Rechenaufwands unterscheiden.

Ein weiterer Schwerpunkt liegt auf der Betrachtung der notwendigen Vorverarbeitungsschritte, wie Tokenisierung und Lemmatisierung, sowie deren Einfluss auf die Qualität der Analyseergebnisse. Auf dieser Grundlage soll ein geeignetes Modell für die praktische Anwendung ausgewählt werden.

Ergänzend wird ein Prototyp einer Webanwendung entwickelt, der es ermöglicht, neue Kundenbewertungen hochzuladen, automatisch zu analysieren und die Ergebnisse in übersichtlicher Form zu visualisieren. Ziel ist es, die praktische Anwendbarkeit der Sentiment-Analyse in einem kundenorientierten Szenario zu demonstrieren.

Optional wird im Rahmen der Arbeit aufgezeigt, inwieweit Sentiment-Analyse grundsätzlich dazu beitragen kann, Hinweise auf mögliche Zusammenhänge zwischen Kundenbewertungen und dem Kaufverhalten zu liefern. Eine konkrete empirische Untersuchung dieses Zusammenhangs konnte jedoch aufgrund fehlender Verkaufsdaten nicht durchgeführt werden und wird daher nicht weiter vertieft.

1.3 Aufbau der Arbeit

Diese Arbeit ist wie folgt aufgebaut. In Kapitel 2 werden die theoretischen Grundlagen der Sentiment-Analyse vorgestellt, um einen grundlegenden Überblick über die verwendeten Konzepte und Methoden zu geben. Kapitel 3 befasst sich mit den Algorithmen und Modellen, die für die Sentiment-Analyse eingesetzt werden können.

In Kapitel 4 werden die verwendeten Datensätze sowie deren Aufbereitung beschrieben. Kapitel 5 erläutert den Modellierungsprozess, einschließlich der eingesetzten Werkzeuge und der Vorgehensweise beim Training der Modelle. In Kapitel 6 werden die erzielten Ergebnisse vorgestellt, evaluiert und miteinander verglichen.

Kapitel 7 beschreibt den Entwurf und die Umsetzung eines Prototyps einer Webanwendung zur Anwendung der Sentiment-Analyse. Abschließend werden in Kapitel 8 die Ergebnisse der Arbeit zusammengefasst und ein Ausblick auf mögliche zukünftige Erweiterungen gegeben.

2 Theoretische Grundlagen

Dieses Kapitel vermittelt die theoretischen Grundlagen, die für das Verständnis der in dieser Arbeit verwendeten Methoden erforderlich sind. Zunächst wird das Konzept der Sentiment-Analyse eingeordnet und in den Kontext des Natural Language Processing (NLP) gestellt. Anschließend werden zentrale Vorverarbeitungsschritte sowie grundlegende Verfahren zur Textrepräsentation erläutert. Abschließend werden wichtige Evaluationsmetriken und Optimierungsansätze vorgestellt, die zur Bewertung der Modellleistung herangezogen werden.

2.1 Grundlagen der Natural Language Processing (NLP)

Natural Language Processing (NLP) ist ein Forschungsgebiet, das sich mit der automatischen Verarbeitung natürlicher Sprache durch Computer beschäftigt. Ziel ist es, Texte oder gesprochene Sprache so aufzubereiten, dass Maschinen deren Inhalt analysieren und interpretieren können. NLP verbindet Methoden aus der Linguistik, Informatik und dem maschinellen Lernen [IBM26c].

Da natürliche Sprache sehr komplex und oft mehrdeutig ist, stellt NLP besondere Anforderungen an Modelle und Algorithmen. Typische Aufgaben im NLP sind unter anderem Tokenisierung, Lemmatisierung, Wortartenerkennung (Part-of-Speech-Tagging) sowie Textklassifikation, beispielsweise im Rahmen der Sentiment-Analyse.

Ein zentrales Problem ist, dass Computer Sprache nicht im menschlichen Sinne verstehen, sondern lediglich statistische Muster erkennen. Deshalb ist eine sorgfältige Vorverarbeitung der Texte besonders wichtig, da die Qualität der Ergebnisse stark davon abhängt [IBM26c].

Ein Beispiel hierfür ist der Satz „Das Produkt ist nicht schlecht“. Trotz des Wortes „schlecht“ muss das Modell erkennen, dass insgesamt eine positive Stimmung ausgedrückt wird.

2.2 Sentiment Analysis

Die Sentiment-Analyse ist ein Teilgebiet des Natural Language Processing (NLP) und befasst sich mit der automatischen Erkennung von Meinungen und Stimmungen in Texten. Ziel ist es, Texte anhand ihrer emotionalen Bewertung zu klassifizieren. Häufig werden dabei die Klassen positiv, negativ und neutral verwendet [IBM26b].

Es existieren verschiedene Ansätze zur Sentiment-Analyse. Lexikonbasierte Methoden nutzen vorab definierte Wortlisten und benötigen keine Trainingsdaten. Machine-Learning-Modelle lernen aus annotierten Beispieldaten. Transformer-Modelle erfassen zusätzlich den Kontext eines Wortes.

Die Aufgabe ist besonders anspruchsvoll, da Sprache ironisch oder mehrdeutig sein kann. Auch Negationen, Ironie oder Sarkasmus stellen eine große Herausforderung dar. Aufgrund dieser Schwierigkeiten ist eine sorgfältige Methodenauswahl notwendig.

2.3 Textvorverarbeitung

2.3.1 Tokenisierung

Unter Tokenisierung versteht man einen grundlegenden Prozess in der Verarbeitung natürlicher Sprache. Dabei wird ein Text in kleine Einheiten zerlegt, die als „Tokens“ bezeichnet werden. Diese Tokens können zum Beispiel einzelne Wörter, Sätze oder Zeichen sein. Welche Art von Tokens verwendet wird, hängt vom jeweiligen Anwendungsfall ab [And24, S. 25].

Die Tokenisierung ist notwendig, um komplexe Texte in eine Form zu bringen, die von einem Computer einfacher verarbeitet und analysiert werden kann. In einfachen Fällen erfolgt die Tokenisierung anhand von Leerzeichen und Satzzeichen. In der Realität ist dies jedoch oft nicht ausreichend, da Sprache Sonderfälle, wie Abkürzungen, Zahlen oder zusammengesetzte Wörter enthalten kann.

Moderne NLP-Systeme verwenden fortgeschrittene Verfahren, zum Beispiel die Subword-Tokenisierung [JM26, Kap. 2, S. 10]. Dabei können Wörter in mehrere Tokens zerlegt werden, was insbesondere bei unbekannten oder seltenen Wörtern von Vorteil ist. Eine fehlerhafte Tokenisierung kann dazu führen, dass wichtige Informationen verloren gehen. Sie ist daher ein zentraler Bestandteil jeder NLP-Pipeline.

Listing 2.1 zeigt die Tokenisierung eines beispielhaften Kundenreviews, bei der der Text in einzelne Wörter und Satzzeichen zerlegt wird.

Listing 2.1: Codebeispiel Python - Tokenisierung

```
1  # Benötigte Bibliothek:
2  # pip install spacy
3  # python -m spacy download de_core_news_sm
4
5      import spacy
6
7      # Deutsches spaCy-Modell laden
8      nlp = spacy.load("de_core_news_sm")
9
10     # Beispiel-Review
11     review = "Das Produkt ist sehr gut verarbeitet und ich bin mit
           dem Kauf zufrieden."
12
13     # spaCy-Dokument erzeugen
```

```
14        doc = nlp(review)
15
16        # Tokenisierung
17        print("Tokenisierung des Reviews:")
18        print([token.text for token in doc])
19
20
21        Tokenisierung des Reviews:
22        ['Das', 'Produkt', 'ist', 'sehr', 'gut', 'verarbeitet', 'und', '
              ich', 'bin', 'mit', 'dem', 'Kauf', 'zufrieden', '.']
```

2.3.2 Lemmatisierung

Die Lemmatisierung bezeichnet den Prozess, Wörter auf ihre lexikalische Grundform, das sogenannte Lemma, zurückzuführen. Ziel ist es, verschiedene Wortformen mit gleicher Bedeutung zusammenzufassen, beispielsweise *running* und *run* oder *ist* und *sein*. Dadurch wird die Anzahl unterschiedlicher Wortformen reduziert [And24, S. 27].

Im Gegensatz zu Stemming, bei dem Wörter oft nur mechanisch gekürzt werden, berücksichtigt die Lemmatisierung grammatikalische Regeln. Dafür ist häufig eine Bestimmung der Wortart notwendig. Die Lemmatisierung verbessert die Vergleichbarkeit von Texten. Allerdings ist sie sprachabhängig und benötigt sprachspezifische Ressourcen [IBM26a].

Listing 2.2 zeigt ein Python-Beispiel, das den Unterschied zwischen Stemming und Lemmatisierung anhand eines deutschen Kundenreviews verdeutlicht.

Listing 2.2: Python Beispiel Lemmatisierung vs. Stemming

```
1         # Benötigte Bibliotheken:
2         # pip install spacy nltk
3         # python -m spacy download de_core_news_sm
4
5         import spacy
6         from nltk.stem import SnowballStemmer
7
8         # Deutsches spaCy-Modell laden
9         nlp = spacy.load("de_core_news_sm")
10
11        # Deutscher Stemmer
12        stemmer = SnowballStemmer("german")
13
14        # Beispielsatz
15        text = "Das Produkt ist sehr gut verarbeitet und ich bin mit dem
              Kauf zufrieden."
16
17        # spaCy-Dokument erzeugen
18        doc = nlp(text)
19
20        print("Originale Tokens:")
21        print([token.text for token in doc])
22
```

```
23          print("\nLemmatisierung:")
24          print([token.lemma_ for token in doc])
25
26          print("\nStemming:")
27          print([stemmer.stem(token.text) for token in doc])
28
29
30          Originale Tokens:
31          ['Das', 'Produkt', 'ist', 'sehr', 'gut', 'verarbeitet', 'und', '
                ich', 'bin', 'mit', 'dem', 'Kauf', 'zufrieden', '.']
32
33          Lemmatisierung:
34          ['der', 'Produkt', 'sein', 'sehr', 'gut', 'verarbeiten', 'und', '
                ich', 'sein', 'mit', 'der', 'Kauf', 'zufrieden', '--']
35
36          Stemming:
37          ['das', 'produkt', 'ist', 'sehr', 'gut', 'verarbeitet', 'und', '
                ich', 'bin', 'mit', 'dem', 'kauf', 'zufried', '.']
```

Die Ausgabe zeigt, dass die Lemmatisierung grammatikalisch korrekte Grundformen erzeugt, während das Stemming lediglich Wortstämme bildet, die nicht immer linguistisch korrekt sind.

2.3.3 Part-of-Speech-Tagging

Part-of-Speech-Tagging (POS-Tagging) beschreibt die automatische Zuordnung von Wortarten zu einzelnen Tokens [And24, S. 47]. Zu den häufigsten Wortarten zählen Nomen, Verben, Adjektive und Adverbien. Diese Information ist wichtig, um die grammatikalische Struktur eines Satzes zu analysieren.

POS-Tags werden häufig als Grundlage für weitere Verarbeitungsschritte genutzt. Sie unterstützen beispielsweise die Lemmatisierung oder die Erkennung wichtiger Wörter. In der Sentiment-Analyse spielen insbesondere Adjektive und Adverbien eine große Rolle, da sie oft für die qualitativen Beschreibung eines Merkmals verwendet werden [NY03]. Fehler im POS-Tagging können sich auf nachfolgende Verarbeitungsschritte auswirken. Im Rahmen der Arbeit werden POS-Tags für die Themaerkennung in Nutzerkommentaren genutzt.

Listing 2.3 zeigt das POS-Tagging eines beispielhaften Kundenreviews. Die Ausgabe besteht aus Tupeln, die jeweils das Token, die zugeordnete Wortart sowie deren Erklärung enthalten.

Listing 2.3: Python Beispiel Part-Of-Speech Tagging

```
1   # Benötigte Bibliotheken:
2   # pip install spacy
3   # python -m spacy download de_core_news_sm
4
5   import spacy
6
7   # Deutsches spaCy-Modell laden
8   nlp = spacy.load("de_core_news_sm")
9
```

```
10        # Beispiel-Review
11        review = "Das Produkt ist sehr gut verarbeitet und ich bin mit
              dem Kauf zufrieden."
12
13        # spaCy-Dokument erzeugen
14        doc = nlp(review)
15
16        # POS-Tagging
17        print("Part-of-Speech-Tagging:")
18        print([(token.text, token.pos_, spacy.explain(token.pos_)) for
              token in doc])
19
20
21        Part-of-Speech-Tagging:
22        [('Das', 'DET', 'determiner'), ('Produkt', 'NOUN', 'noun'), ('ist
              ', 'AUX', 'auxiliary'), ('sehr', 'ADV', 'adverb'), ('gut', '
              ADV', 'adverb'), ('verarbeitet', 'VERB', 'verb'), ('und', '
              CCONJ', 'coordinating conjunction'), ('ich', 'PRON', 'pronoun'
              ), ('bin', 'AUX', 'auxiliary'), ('mit', 'ADP', 'adposition'),
              ('dem', 'DET', 'determiner'), ('Kauf', 'NOUN', 'noun'), ('
              zufrieden', 'ADV', 'adverb'), ('.', 'PUNCT', 'punctuation')]
```

2.4 TF-IDF

Die Abkürzung TF-IDF steht für *Begriffshäufigkeit - inverse Dokumentenhäufigkeit* (engl. „Term-Frequency - Inverse Data Frequency“). Das ist ein Verfahren zur numerischen Repräsentation von Texten. Dabei wird gemessen, wie häufig ein Wort in einem Dokument vorkommt und wie selten ist es in gesamte Datensammlung. Wörter, die in vielen Dokumenten vorkommen, erhalten ein geringeres Gewicht und ein Wort, das in sehr wenigen Dokumenten erscheint, bekommt einen höheren Gewicht [Ali19, S. 61–62].

Ziel von TF-IDF ist es, besonders aussagekräftige Begriffe hervorzuheben [Ali19, S. 62]. TF-IDF ignoriert den Kontext der Wörter und betrachtet sie unabhängig voneinander. Dennoch wird es häufig eingesetzt, da es einfach, effizient und gut interpretierbar ist. In Kombination mit klassischen Modellen wie logistischer Regression oder Support Vector Machine liefert TF-IDF oft solide Ergebnisse.

Listing 2.4 zeigt, wie Kundenreviews mithilfe von TF-IDF in numerische Vektoren umgewandelt werden. Das Verfahren gewichtet dabei seltene, aber inhaltlich relevante Wörter stärker als häufig vorkommende Begriffe und ermöglicht so eine effektive Textklassifikation.

Listing 2.4: Python Beispiel TF-IDF

```
1         # Benötigte Bibliothek:
2         # pip install scikit-learn
3
4         from sklearn.feature_extraction.text import TfidfVectorizer
5         import pandas as pd
6
7         # Beispiel-Reviews
```

```
8         reviews = [
9         "Das Produkt ist sehr gut und hochwertig verarbeitet",
10        "Ich bin mit dem Kauf sehr zufrieden",
11        "Die Qualität ist schlecht und enttäuschend"
12        ]
13
14        # TF-IDF-Vektorisierer initialisieren
15        vectorizer = TfidfVectorizer()
16
17        # TF-IDF-Matrix berechnen
18        tfidf_matrix = vectorizer.fit_transform(reviews)
19
20        # In DataFrame umwandeln (besser lesbar)
21        tfidf_df = pd.DataFrame(
22                        tfidf_matrix.toarray(),
23                        columns=vectorizer.get_feature_names_out()
24                )
25
26        print("TF-IDF-Matrix:")
27        print(tfidf_df)
28
29
30           bin       das       dem       ...   und       verarbeitet
                zufrieden
31        0  0.000000  0.385323  0.000000  ...  0.293048     0.385323
             0.000000
32        1  0.389888  0.000000  0.389888  ...  0.000000     0.000000
             0.389888
33        2  0.000000  0.000000  0.000000  ...  0.334907     0.000000
             0.000000
34
35        [3 rows x 18 columns]
```

2.5 Evaluationsmetriken

2.5.1 Precision

Die Precision, auch Präzision genannt, bezeichnet das Verhältnis der korrekt vorhergesagten positiven Fälle zu allen als positiv klassifizierten Fällen [Aur18, S. 87].

$$\textit{Präzision} = \frac{TP}{TP + FP} \tag{2.1}$$

Dabei steht die TP (True Positive) steht für korrekt als positiv vorhergesagte Fälle, während FP (False Positive) Fälle bezeichnet, die fälschlicherweise als positiv klassifiziert wurden, obwohl sie tatsächlich negativ sind.

2.5.2 Recall

Recall, auch Sensitivität genannt, beschreibt das Verhältnis der korrekt vorhergesagten positiven Fälle zur Gesamtanzahl der tatsächlich positiven Fälle [Aur18, S. 87].

$$\textit{Recall} = \frac{TP}{TP + FN} \tag{2.2}$$

Hierbei bezeichnet FN (False Negative) Fälle, die fälschlicherweise als negativ vorhergesagt wurden.

2.5.3 F1-Score

F1-Score ist das harmonische Mittel aus Precision und Recall und stellt ein ausgewogenes Maß zur Bewertung der Modellleistung dar [Aur18, S. 88]. Er ist besonders geeignet, wenn ein Gleichgewicht zwischen Precision und Recall erwünscht wird.

$$\textit{F1} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.3}$$

2.6 Hyperparameter

Hyperparameter sind Modellparameter, die vor dem Training festgelegt werden und das Lernverhalten eines Modells steuern. Beispiele hierfür sind der Regularisierungsparameter bei der logistischen Regression oder die Anzahl der Entscheidungsbäume in einem Random-Forest-Modell.

Die Wahl geeigneter Hyperparameter hat einen großen Einfluss auf die Modellleistung. Eine falsche Einstellung kann zu Über- oder Unteranpassung führen. Da eine manuelle Auswahl oft ineffizient ist, werden automatisierte Verfahren eingesetzt.

Ein verbreiteter Ansatz ist die Random Search, bei der zufällige Kombinationen von Hyperparametern innerhalb definierter Bereiche getestet werden, anstatt alle möglichen Kombinationen zu prüfen. Diese Methode ist besonders effizient bei großen Suchräumen und liefert häufig vergleichbare Ergebnisse zu *GridSearchCV*, jedoch mit deutlich geringerem Zeit- und Rechenaufwand [Aur18, S. 73, 75].

3 Stand der Technik

In diesem Kapitel werden die Ansätze für die Sentiment Analyse vorgestellt. Und zwar werden zuerst die klassischen Machine-Learning-Modelle beschrieben. Danach werden die lexikonbasierten Ansätze vorgestellt. Abschließend werden die Transformer-Modelle beschrieben.

3.1 Klassische Machine-Learning-Ansätze

3.1.1 Logistische Regression

Die Logistische Regression ist ein lineares Modell, das die Beziehung zwischen einer oder mehreren unabhängigen Variablen und einer Zielvariable untersucht.

Sie wird hauptsächlich für Klassifikationsprobleme verwendet, bei denen das Ziel ist, eine Entscheidung zwischen zwei Kategorien zu treffen [Aur18, S. 135]. Zum Beispiel könnte das Modell verwendet werden, um zu entscheiden, ob ein Nutzerkommentar positiv oder negativ ist. Das Modell berechnet eine Wahrscheinlichkeit für jede mögliche Klasse. Der Ausgabewert liegt im Bereich zwischen 0 und 1. Diese Wahrscheinlichkeit wird durch die sogenannte Sigmoid-Funktion berechnet.

Wenn das Modell für mehr als zwei Kategorien entscheiden muss, wird es auf multinomiale logistische Regression erweitert, wobei für jede Kategorie eine eigene Wahrscheinlichkeit berechnet wird. Die Kategorie mit der höchsten Wahrscheinlichkeit wird als Vorhersage ausgewählt.

3.1.2 Support Vector Machine

Ein weiterer Ansatz im Machine Learning ist die Support Vector Machine (SVM). Diese wird sowohl für Klassifikationsprobleme (Support Vector Classifier), als auch für Regressionsprobleme (Support Vector Regressor) verwendet [Aur18, S. 145]. Im Rahmen dieser Arbeit wird der Support Vector Classifier betrachtet.

Das Modell versucht eine optimale Linie beziehungsweise Hyperebene zwischen den Klassen zu finden, welche den Abstand zu den nächstliegenden Datenpunkten maximiert [Aur18, S. 145]. Abhängig von der Anzahl der Eingabeparameter stellt die Hyperebene eine Linie in 2-D-Raum oder eine Ebene in n-dimensionalen Raum dar. Da es mehrere Möglichkeiten gibt den Datensatz zu trennen, ermöglicht die Maximierung dieses Abstands die Bestimmung einer bestmöglichen Entscheidungsgrenze.

Die Support Vector Machine ist ein weit verbreiteter Algorithmus in Machine Learning, da sie sowohl für linear- als auch für nicht-linear trennbare Daten eingesetzt werden kann. Dabei kommen die sogenannten Kernel-Funktionen zum Einsatz, welche die Daten in einen höherdimensionalen Raum transformieren, um eine lineare Trennung zu ermöglichen [Aur18, S. 149].

Die Wahl der Kernel-Funktion, wie beispielsweise der lineare Kernel, der Polynom-Kernel, die Radial Basis Function(RBF) oder der Sigmoid-Kernel, hängt von den Daten und dem jeweiligen Anwendungsfall ab.

3.1.3 Random Forest

Random Forest ist ein überwachter Lernen Ansatz im Machine Learning, der sowohl für Klassifikations- als auch für Regressionsprobleme eingesetzt wird. Das Verfahren basiert auf einer Vielzahl von Entscheidungsbäumen, deren Anzahl als Modellparameter festgelegt wird.

Jeder Entscheidungsbaum wird mit einem zufällig gezogenen Teil des Trainingsdatensatzes trainiert, wobei die Ziehung mit Zurücklegen erfolgt (Bootstrap-Sampling). Zusätzlich wird bei jeder Aufteilung eines Knotens nur eine zufällige Auswahl der verfügbaren Merkmale berücksichtigt [Aur18, S. 189]. Dadurch unterscheiden sich die einzelnen Bäume voneinander. Innerhalb eines Baumes werden die Daten anhand bestimmter Bedingungen aufgeteilt.

Zur Bewertung der Qualität eines Splits können Metriken, wie Gini-Verunreinigung für Klassifikationsprobleme oder mittlere Quadratische Abweichung (Mean Squared Error) für Regressionsprobleme verwendet werden. Abhängig vom Ergebnis der Bedingung folgen die Daten entweder dem „Ja“-Zweig oder dem alternativen Pfad.

Die Aufteilung der Daten läuft solange, bis eine eindeutige Klassenzugehörigkeit erfolgt oder eine vordefinierte maximale Baumtiefe erreicht wird. Befinden sich in einem Endknoten mehrere Klassen, wird die Klasse mit der höchsten Häufigkeit ausgewählt.

Nach dem Training werden neue Datensätze einzeln durch alle Bäume geleitet. Jeder Baum gibt eine eigene Vorhersage ab und die endgültige Klassenzuordnung erfolgt durch Mehrheitsentscheidung. Die Kombination vieler unterschiedlicher Bäume liefert in der Praxis robuste Ergebnisse und das Modell ist weniger anfällig für Overfitting.

3.2 Lexikonbasierte Verfahren

3.2.1 VADER Sentiment

VADER (Valence Aware Dictionary and sEntiment Reasoner) ist ein lexikon- und regelbasiertes Verfahren für die Sentiment Analyse, die speziell entwickelt wurde, um die Stimmung in kurzen informellen Texten, wie Social-Media-Beiträge zu bestimmen. Das Modell basiert auf einer vordefinierten Wortliste, in der jedem Wort ein Sentimentwert zugeordnet ist [HG14].

Die Autoren betonen, dass das Modell keine tiefere sprachliche Analyse durchführt. Stattdessen werden heuristische Regeln verwendet, um Effekte, wie Negation oder Verstärkung abzubilden. Zum Beispiel verändert das Wort „not“ die Polarität (Stimmung) eines nachfolgenden positiven Begriffs. Auch Satzzeichen, wie Ausrufezeichen oder Großschreibung fließen in die Bewertung ein. Die Regeln wurden manuell entworfen und getestet.

VADER benötigt kein Training auf eigenen Daten, da keine Modellanpassung erfolgt. Die Gesamtbewertung erfolgt durch die Summation der Wortwerte zu einem Compound Score der anschließend auf Werte zwischen -1 und $+1$ normiert wird. Folgende Sentimentwerte sind möglich [Hut26]:

- positives Sentiment (Compound Score >= 0.05)
- neutrales Sentiment (-0.05 <= Compound Score <= 0.05)
- negatives Sentiment (Compound Score <= -0.05)

Allerdings ist das Modell stark von der Qualität des Lexikons abhängig. Für längere oder komplexere Texte nimmt die Zuverlässigkeit ab. VADER ist vor allem für englischsprachige Texte geeignet und nur eingeschränkt auf andere Sprachen übertragbar.

3.2.2 TextBlob

TextBlob ist eine Python-Bibliothek, die grundlegende Funktionen zur Verarbeitung natürlicher Sprache bereitstellt. Dazu gehören unter anderem Tokenisierung, Part-of-Speech-Tagging sowie eine Sentiment-Analyse für englischsprachige Texte. Die Sentiment-Analyse basiert auf der Zuweisung von Wortpolaritäten aus einem vordefinierten Lexikon.

Für einen gegebenen Text berechnet TextBlob einen Polaritätswert, der angibt, ob der Text eher positiv oder negativ ist, sowie einen Subjektivitätswert, der den Grad der Meinungsäußerung beschreibt. Diese Berechnung erfolgt ohne Modelltraining und ausschließlich auf lexikonbasierter Grundlage.

TextBlob ist primär für die englische Sprache konzipiert. Zwar existieren Erweiterungen und inoffizielle Ansätze zur Nutzung mit anderen Sprachen, jedoch werden diese nicht systematisch dokumentiert. Aussagen zur Leistungsfähigkeit in komplexen sprachlichen Kontexten, wie Ironie oder Sarkasmus, werden in der offiziellen Dokumentation nicht explizit getroffen und können daher nicht belegt werden [Lor25].

3.3 Transformer-basierte Modelle

3.3.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) ist ein transformerbasiertes Sprachmodell, das von Devlin et al. (2019) vorgestellt wurde.

Im Gegensatz zu klassischen Ansätzen der Natural Language Processing, verarbeitet BERT Text bidirektional, sodass jedes Wort sowohl seinen linken als auch rechten Kontext

berücksichtigt. Das Modell wird zunächst auf größere Textkorpora wie BooksKorpus (800M Wörter) und englische Wikipedia (2.500M Wörter) vortrainiert. Dabei kommen die Aufgaben Masked Language Modelling und Next Sentence Prediction zum Einsatz.

Für konkrete Aufgaben wie zum Beispiel Sentiment Analysis ist eine Feinjustierung notwendig, weil die Anpassungsfähigkeit des Modells begrenzt bleiben kann, was potentiell die Vorhersagen im Sentiment Analysis beeinflusst [Dev+19].

In der Sentiment-Analyse ermöglicht diese Kontextsensitivität eine detaillierte Interpretation von Aussagen, insbesondere bei komplexen Satzstrukturen. BERT erfordert jedoch hohen Rechenaufwand und spezialisierte Hardware und kann für einfachere Klassifikationsaufgaben sehr kompliziert sein.

3.3.2 RoBERTa

RoBERTa (Robustly optimized BERT approach) ist eine Weiterentwicklung von BERT und basiert auf derselben Transformer-Architektur. Der Unterschied liegt im Trainingsprozess. Die Autoren verzichten auf Next Sentence Prediction und nutzen größere Datenmengen. Dadurch kann das Modell stabiler trainiert werden. Die grundlegende Funktionsweise bleibt jedoch identisch zu BERT. Die Texte werden auch als Token-Sequenzen verarbeitet.

RoBERTa wird häufig für Textklassifikation und Sentiment-Analyse eingesetzt. Durch einen verbesserten Trainingsprozess könnte die RoBERTa in vielen Benchmarks bessere Ergebnisse als BERT erzielen. Auch hier ist Feinjustierung für konkrete Aufgaben erforderlich. Der Ressourcenbedarf ist mindestens so hoch wie bei BERT [Zhu+21].

3.3.3 German Sentiment

German Sentiment ist ein Transformer-basiertes Modell für die Sentiment-Analyse deutschsprachiger Texte. Es basiert auf einer BERT-ähnlichen Architektur, die den bidirektionalen Kontext von Wörtern nutzt, um die Bedeutung von Texten zu erfassen. Das Modell wurde gezielt für die Klassifikation von Texten in positiv, neutral und negativ entwickelt und ist somit auf Sentiment-Analysen spezialisiert [Guh+20].

Für das Training wurden mehrere deutschsprachige Datensätze aus unterschiedlichen Bereichen genutzt, um eine möglichst breite Abdeckung verschiedener Textarten zu erreichen.

4 Daten und Vorverarbeitung

Kapitel 4 beschäftigt sich mit der Vorstellung der verwendeten Datensätze für das Training und die Datenvorverarbeitung.

4.1 Verwendete Datensätze

Für die Durchführung der Sentiment-Analyse werden in dieser Arbeit zwei Datenquellen verwendet: für das Training der Modelle wurde der Datensatz „Amazon reviews: Clothing_Shoes_and_Jewelry_5.json“ der Stanford University ausgewählt und für die kleine Webanwendung wurden Rezensionen von drei Produkten der „Firma X“ aus dem Zeitraum von 2020 bis 2025 erhoben. Die Datenerhebung erfolgte unter Berücksichtigung der Datenschutz-Grundverordnung (DSGVO).

Der Amazon-Datensatz umfasst insgesamt 11.285.464 Bewertungen und enthält alle notwendigen Informationen, wie den Bewertungstext und die zugehörige Sternbewertung. Die für die Webanwendung gesammelten Produktrezensionen umfassen 102, 135 und 257 Reviews. Beide Datensätze lagen im JSON-Format vor und wurden in die Pandas-DataFrames umgewandelt. Für Trainingszwecke wurden 90.000 Rezensionen zufällig aus dem Amazon-Datensatz ausgewählt, wobei folgende Voraussetzungen erfüllt sein müssen:

1. Die ganzzahligen Sternbewertungen wurden in Sentiment-Klassen überführt (1 und 2 Sterne $\rightarrow$ negativ, 3 $\rightarrow$ neutral, 4 und 5 $\rightarrow$ positiv).
2. Zur Sicherstellung des Klassengleichgewichts wurden jeweils 30.000 positive, 30.000 neutrale, 30.000 negative Rezensionen ausgewählt.
3. Die Länge der Rezensionen musste zwischen 10 und 200 Wörtern liegen. Sehr kurze Texte enthalten häufig zu wenig Information, während sehr lange Texte viele irrelevante Inhalte enthalten können, was die Modellleistung negativ beeinflussen kann.

Eine solche Verteilung ist wichtig für die Auswahl geeigneter Evaluationsmetriken, da eine ungleiche Klassenverteilung die Aussagekraft bestimmter Metriken negativ beeinflussen kann.

Da der verwendete Datensatz ausschließlich englischsprachige Rezensionen enthält, und sich diese Arbeit auch auf den deutschsprachigen Raum bezieht, wurde entschieden, die Hälfte der ausgewählten Daten ins Deutsche zu übersetzen. Eine Suche nach einem vergleichbaren deutschsprachigen Datensatz verlief erfolglos. Entweder waren die verfügbaren Datensätze zu klein (etwa 500 Einträge) oder sie enthielten Rezensionen in mehreren Sprachen, wobei der Anteil deutschsprachiger Texte sehr gering war.

4.1.1 Kritische Reflexion der Übersetzung

Die Übersetzung kann zu Veränderungen in der Bedeutung von Texten führen, insbesondere bei Umgangssprache, Ironie oder kürzeren Sätzen. Daher wird die übersetzte Teilmenge nicht als vollständiger Ersatz für deutschsprachige Rezensionen betrachtet, sondern dient als ergänzende Datenbasis. Die Ergebnisse der Sentiment-Analyse müssen unter Berücksichtigung dieser Einschränkungen interpretiert werden.

4.2 Datenaufteilung

Beide Teildatensätze wurden im Verhältnis 75% Trainingsdaten und 25% Testdaten aufgeteilt. Die Aufteilung erfolgte zufällig, aber unter Berücksichtigung der Klassenverteilung (stratifizierte Aufteilung), um sicherzustellen, dass alle Teilmengen ähnliche Anteile der Bewertungen enthalten. Dadurch werden die Testdaten nicht im Trainingsprozess verwendet, was die bessere Einschätzung der Modellleistung ermöglicht [JM26, Kap. 3, S. 8].

4.3 Konkrete Vorverarbeitungsschritte

Die Textvorverarbeitung wird abhängig vom jeweiligen Modell durchgeführt. Klassische Machine-Learning-Modelle erfordern umfangreiche Vorverarbeitungsschritte, während moderne Transformer-Modelle weniger Vorverarbeitung benötigen.

4.3.1 Klassische Modelle

Für die Modelle Logistische Regression, Support Vector Machine und Random Forest Classifier werden die im Kapitel 2 beschriebenen NLP-Schritte konkret auf den verwendeten Datensatz angewendet. Ziele der Vorverarbeitung:

1. Texttokenisierung
2. Umwandlung in Kleinbuchstaben
3. Textlemmatisierung zur Reduktion auf Wortgrundformen
4. Entfernung von Stoppwörtern, außer den Wörtern, die für Bezeichnung der negativen Kontext notwendig sind (zum Beispiel: kein, nicht, nie, never, not usw.)
5. Umwandlung der Texte in numerische Vektoren mittels TF-IDF

Diese Schritte sorgen dafür, dass irrelevante Informationen entfernt und die Texte in eine einheitliche Form gebracht werden. Auf diese Weise können die Modelle die für die Sentiment Analyse wichtiger Wörter und Phrasen effizienter erkennen und verarbeiten.

4.3.2 Vorverarbeitung für lexikonbasierte Modelle

Lexikonbasierte Modelle wie TextBlob und VADER benötigen nur eine minimale Vorverarbeitung. Die Modelle arbeiten direkt auf dem Rohtext, da sie eigene Regeln und Lexika verwenden. Eine starke Vorverarbeitung könnte in diesen Fall zu einem Informationsverlust führen.

4.3.3 Vorverarbeitung für Transformer-Modelle

Transformer-basierte Modelle wie BERT, RoBERTa und German Sentiment verwenden eigene Tokenizer. Daher werden die Texte lediglich bereinigt, jedoch nicht lemmatisiert oder in TF-IDF-Vektoren umgewandelt. Die Modelle verarbeiten den Rohtext direkt, wodurch semantische Zusammenhänge und Wortkontexte erhalten bleiben.

5 Modellierung und Implementierung

In diesem Kapitel werden die eingesetzten Werkzeuge, der Trainingsprozess, die Hyperparameter-Optimierung sowie die Auswahl der Modelle beschrieben.

5.1 Implementierungsumgebung und Werkzeuge

Die Implementierung erfolgt in der Programmiersprache *Python*, da diese Sprache eine große Anzahl von Bibliotheken für Machine Learning und Natural Language Processing bietet. Folgende Werkzeuge werden im Rahmen dieser Arbeit verwendet:

- *pandas* - Datenverarbeitung und -analyse
- *scikit-learn* - klassische Machine Learning Modelle, TF-IDF-Vektorisierung und RandomSearchCV
- *spaCy* - linguistische Vorverarbeitung
- *SQLAlchemy* - Datenbankanbindung
- *Flask* - Erstellung einer einfachen Webanwendung
- *matplotlib* - Visualisierung der Ergebnisse
- *transformers* - Implementierung von BERT- und RoBERTa-Modellen
- *gensim* - Bestimmung dominanter Themen in Reviews
- *Docker* - Containerisierung der Anwendung und Sicherstellung einer konsistenten Laufzeitumgebung
- *DBeaver* - Verwaltung und Analyse der Datenbank

Docker und *DBeaver* wurden insbesondere in der frühen Projektphase eingesetzt, da ursprünglich die Umsetzung einer Java-basierten Anwendung vorgesehen war. Die Nutzung von Docker erleichterte dabei die flexible Anpassung der technischen Architektur.

5.2 Versuchsaufbau und Vergleichsstrategie

Der Trainingsprozess unterscheidet sich je nach Modelltyp:

- Klassische Machine-Learning-Modelle werden auf TF-IDF-Vektoren trainiert
- Lexikonbasierte Modelle und Transformer-Modelle arbeiten direkt mit Rohtext

Für alle Modelle wird ein einheitlicher Evaluationsprozess angewendet, um einen fairen Vergleich zu gewährleisten. Die Modelle werden ausschließlich mit den Trainingsdaten trainiert, während der Testdatensatz zur Bestimmung des F1-Scores verwendet wird. Zur Sicherstellung reproduzierbarer Ergebnisse wird beim Aufteilen des Datensatzes ein fester Zufalls-Seed (*random_state*) verwendet [Sci26].

Für klassische Machine-Learning-Modelle wird eine Hyperparameter-Optimierung mittels *RandomSearchCV* in Kombination mit Kreuzvalidierung durchgeführt.

Optimiert werden unter anderem:

- Support Vector Machine: Kernelfunktion, Regularisierungsparameter und Kernelkoeffizient für Kernel-Funktionen
- Logistische Regression: Regularisierungsparameter, Solver-Algorithmus
- Random Forest: Anzahl der Entscheidungsbäume, Mindestanzahl an Trainingsbeispiele pro Knoten, Anzahl der Merkmale für den besten Split

Lexikonbasierte Modelle benötigen kein Training und werden direkt auf dem Testdatensatz angewendet. Die Trennung in Trainings- und Testdaten bleibt dennoch bestehen, um eine konsistente und vergleichbare Evaluation aller Modelltypen zu gewährleisten.

Transformer-Modelle werden mit vortrainierten Gewichten verwendet und nicht zusätzlich feinjustiert, da die Feinjustierung den zeitlichen und rechnerischen Rahmen der Arbeit überschritten hätte. Ziel war ein fairer Vergleich unter praxisnahen Bedingungen.

Verglichen werden die Modelltypen hinsichtlich Vorhersagequalität (F1-Score) und Rechenaufwand (Trainingszeit). Die transformerbasierten Modelle BERT und RoBERTa wurden aufgrund ihres hohen Rechenbedarfs auf einer GPU ausgeführt, während alle übrigen Modelle auf der CPU trainiert wurden. Die gemessenen Trainingszeiten dienen ausschließlich dazu, den Rechenaufwand der Modelle grob zu vergleichen. Sie sind nicht als exakter, hardwareunabhängiger Leistungsvergleich zu verstehen.

Bei der Trainingszeit handelt es sich um die tatsächlich aufgebrachte Rechenzeit für einen einmaligen Trainingsdurchlauf pro Modell. Jedes Modell wird genau einmal trainiert. Ein erneutes Training erfolgt nur, falls kein gültiges Ergebnis vorliegt. Mehrfache Trainingsläufe wurden bewusst nicht durchgeführt, da der Fokus auf dem Vergleich unterschiedlicher Modellklassen lag und nicht auf der Varianz einzelner Initialisierungen. Die Ergebnisse des Modellvergleichs werden in Kapitel 6 vorgestellt.

Für einen reibungslosen Ablauf der Experimente wurden zwei separate *JSON*-Dateien erstellt:

- *config.json*: Enthält die Hyperparameter, die Sprache sowie den verwendeten Pipeline-Typ (Art der Textvorverarbeitung) für jedes Modell
- *result.json*: Speichert die erzielten Ergebnisse

Dieses Vorgehen ermöglicht es, bereits erfolgreich getestete Modelle im Fehlerfall nicht erneut trainieren zu müssen, wodurch Rechenzeit eingespart wird.

Nach dem Training wird für jede Sprache jeweils ein Modell ausgewählt, das den höchsten F1-Score bei gleichzeitig akzeptabler Laufzeit erzielt. Diese Modelle sollen anschließend für die finale Evaluation verwendet werden.

6 Evaluation und Ergebnisse

In diesem Kapitel werden die Ergebnisse der durchgeführten Sentiment-Analyse vorgestellt und ausgewertet. Die Modelle wurden getrennt für die deutsche und englische Sprache evaluiert. Als Hauptbewertungsmetrik wurde der F1-Score verwendet. Zusätzlich wurde die Ausführungszeit gemessen, um die Effizienz der Modelle zu vergleichen.

6.1 Ergebnisse der Modellvergleiche für deutsche Texte

6.1.1 Klassische Machine-Learning-Modelle

In Tabelle 6.1 werden die Ergebnisse für die klassischen Machine-Learning-Modelle für die deutsche Sprache dargestellt. Die betrachteten Modelle zeigen eine ähnliche, insgesamt mittlere F1-Leistung im Bereich von 0.62 bis 0.63. Deutlich unterscheiden sich hingegen die Laufzeiten der Modelle.

Modell	F1	Dauer (s)	Verwendete Hardware
Logistische Regression	0.63	29.62	CPU
Support Vector Machine	0.63	1954.2	CPU
Random Forest	0.62	441.32	CPU

Tabelle 6.1: Klassische Machine-Learning-Modelle - deutsch

Insbesondere die Support Vector Machine weist eine wesentlich längere Ausführungszeit auf, was in direktem Zusammenhang mit der Verwendung von Kernel-Funktionen steht, die eine höhere rechnerische Komplexität verursachen.

Die Ergebnisse deuten darauf hin, dass klassische Machine-Learning-Ansätze bei geeigneter Parametrisierung der TF-IDF-Vektorisierung und einer strikteren Textvorverarbeitung eine verbesserte Leistung erzielen können. Dazu zählt unter anderem das Entfernen von Wörtern, die nur sehr selten im Dokumentenkorpus auftreten (zum Beispiel weniger als 5-10 Vorkommen), sowie von Wörtern, die fast in allen Dokumenten enthalten sind, wie etwa Artikeln oder Präpositionen. Die sorgfältige Textvorverarbeitung ist ein wichtiger Schritt, da klassische Modelle stark von der Qualität der Eingabedaten abhängen.

Unter diesen Voraussetzungen können klassische Modelle eine konkurrenzfähige Alternative zu komplexeren Modellen darstellen, insbesondere in Szenarien, in denen begrenzte Rechenressourcen zur Verfügung stehen oder der Einsatz spezialisierter Hardware nicht möglich ist.

6.1.2 Lexikonbasierte Modelle

Die Tabelle 6.2 zeigt die Ergebnisse von lexikonbasierten Modelle. Insgesamt fallen die erzielten F1-Werte niedriger aus als erwartet. Besonders VADER erreicht mit einem F1-Score von 0.32 eine geringe Klassifikationsleistung, während TextBlob mit einem Wert von 0.40 nur leicht bessere Ergebnisse erzielt. Auffällig ist zudem, dass sich die beiden Modelle deutlich in ihrer Laufzeit unterscheiden, obwohl beide auf einem lexikonbasierten Ansatz beruhen.

Modell	F1	Dauer (s)	Verwendete Hardware
VADER	0.32	15.05	CPU
TextBlob	0.4	2459.01	CPU

Tabelle 6.2: Lexikonbasierte Modelle - deutsch

Die vergleichsweise schwache Leistung lässt sich dadurch erklären, dass lexikonbasierte Modelle ausschließlich auf vordefinierten Wortlisten basieren und keine Lernphase durchlaufen. Dadurch können sie den Kontext eines Textes nur eingeschränkt erfassen. Zudem wurde VADER, wie im Abschnitt 3.2.1 beschrieben, für kurze und informelle Texte aus sozialen Medien entwickelt, während TextBlob nur eine begrenzte Unterstützung für die deutsche Sprache bietet. Diese Faktoren können die niedrigen F1-Werte in diesem Anwendungsfall erklären.

6.1.3 Transformer-Modelle

Tabelle 6.3 zeigt die Ergebnisse der evaluierten Transformer-basierten Modelle. Das beste Resultat erzielte RoBERTa mit einem F1-Score von 0.65, gefolgt vom Vorgängermodell BERT mit einem Wert von 0.63.

Auffällig ist das vergleichsweise niedrige Ergebnisse von German Sentiment, das mit einem F1-Score von 0.51 deutlich hinter den Erwartungen zurückbleibt, obwohl das Modell speziell für die deutsche Sprache entwickelt wurde.

Die deutlichen Unterschiede zwischen den Ausführungszeiten lassen sich unter anderem durch den Einsatz spezieller Hardware im Fall von BERT und RoBERTa erklären.

Modell	F1	Dauer (s)	Verwendete Hardware
BERT	0.63	436.07	GPU
RoBERTa	0.65	463.6	GPU
German Sentiment	0.51	7040.83	CPU

Tabelle 6.3: Transformer-Modelle - deutsch

Die moderaten Ergebnisse von BERT und RoBERTa lassen sich unter anderem dadurch erklären, dass beide Modelle im Rahmen dieser Arbeit nicht weiter feinjustiert (Fine-Tuning) wurden. Zudem war der verwendete Trainingsdatensatz mit etwa 45.000 Texten relativ klein, was die Anpassungsfähigkeit der Modelle an den spezifischen Anwendungsfall einschränken kann.

Im Fall von German Sentiment ist zu berücksichtigen, dass das Modell als vorgefertigtes Sentiment-Modell ohne weiteres Training eingesetzt wurde. Unterschiede zwischen den zugrunde liegenden Trainingsdaten des Modells und dem in dieser Arbeit verwendeten Datensatz könnten daher zu einer eingeschränkten Übertragbarkeit und somit zu den beobachteten Leistungssenkung geführt haben.

Darüber hinaus verdeutlichen diese Ergebnisse, dass auch vortrainierte Modelle nicht zwangsläufig eine bessere Performance erzielen, wenn sie auf Datensätze angewendet werden, die sich strukturell oder inhaltlich von den ursprünglichen Trainingsdaten unterscheiden.

6.2 Ergebnisse der Modellvergleiche für englische Texte

6.2.1 Klassische Machine-Learning-Modelle

Tabelle 6.4 fasst die Ergebnisse der klassischen Machine-Learning-Modelle zusammen, die auf englischsprachigen Texten getestet wurden. Im Vergleich zu den deutschen Datensätzen erreichen alle drei Modelle leicht höhere F1-Werte, wobei die Unterschiede zwischen den einzelnen Verfahren gering ausfallen.

Modell	F1	Dauer (s)	Verwendete Hardware
Logistische Regression	0.65	15.2	CPU
Support Vector Machine	0.66	2287.58	CPU
Random Forest	0.65	356.28	CPU

Tabelle 6.4: Klassische Machine-Learning-Modelle - englisch

Die insgesamt kürzeren Ausführungszeiten – mit Ausnahme der Support Vector Machine – könnten auf Unterschiede in der Textvorverarbeitung zurückzuführen sein, insbesondere auf eine geringere Anzahl extrahierter Tokens. Eine reduzierte Tokenanzahl führt in der Regel zu einer niedrigeren dimensionalen Repräsentation der Texte und somit zu einer geringeren Rechenkomplexität.

Insgesamt zeigen die Ergebnisse, dass klassische Machine-Learning-Modelle bei Anwendung identischer Vorverarbeitungsschritte eine vergleichbare Leistungsfähigkeit unabhängig von der betrachteten Sprache aufweisen. Dies deutet darauf hin, dass der Einfluss der Sprache bei diesen Modellen geringer ist als bei komplexeren, sprachspezifischen Ansätzen.

6.2.2 Lexikonbasierte Modelle

Die Tabelle 6.5 zeigt die Ergebnisse der lexikonbasierten Modelle für die englische Sprache. Beide Modelle erzielen F1-Werte im unteren Bereich, wobei TextBlob mit einem F1-Score von 0.44 leicht bessere Ergebnisse erreicht als VADER mit einem Wert von 0.42.

Im Vergleich zum deutschen Datensatz fallen die Laufzeiten für beide Modelle deutlich kürzer aus. Dies lässt sich unter anderem dadurch erklären, dass sowohl VADER als auch

Modell	F1	Dauer (s)	Verwendete Hardware
VADER	0.42	15.66	CPU
TextBlob	0.44	20.41	CPU

Tabelle 6.5: Lexikonbasierte Modelle - englisch

TextBlob ursprünglich für die Verarbeitung englischsprachiger Texte entwickelt wurden und daher besser auf diese Sprache abgestimmt sind.

Die Ergebnisse verdeutlichen, dass lexikonbasierte Modelle zwar effizient und schnell einsetzbar sind, ihre Leistungsfähigkeit jedoch durch die feste Wortlistenstruktur begrenzt bleibt. Sprachliche Besonderheiten wie Mehrdeutigkeiten oder kontextabhängige Bedeutungen können nur eingeschränkt berücksichtigt werden, was sich in den vergleichsweise niedrigen F1-Werten widerspiegelt.

6.2.3 Transformer-Modelle

Tabelle 6.6 stellt die Ergebnisse der Transformer-basierten Modelle für den englischsprachigen Datensatz dar. Insgesamt zeigen die Modelle eine höhere Leistungsfähigkeit als in der deutsche Variante, wobei RoBERTa mit einem F1-Score von 0.75 das beste Ergebnis erzielt. BERT folgt mit einem F1-Score von 0.73 und weist damit eine vergleichbare Perfomance auf.

Modell	F1	Dauer (s)	Verwendete Hardware
BERT	0.73	426.75	GPU
RoBERTa	0.75	462.96	GPU

Tabelle 6.6: Transformer-Modelle - englisch

Die Ergebnisse deuten darauf hin, dass beide Modelle in der Lage sind, englischsprachige Texte zuverlässig zu klassifizieren. Der geringe Leistungsunterschied zwischen BERT und RoBERTa kann durch Verbesserungen in der Modellarchitektur und im Vortraining von RoBERTa erklärt werden.

Da auch für die englische Sprache kein zusätzliches Fine-Tuning der Modelle durchgeführt wurde, zeigen die Ergebnisse die Leistung der vortrainierten Modelle ohne weitere Anpassung an den Datensatz. Die insgesamt höheren F1-Werte im Vergleich zur deutschen Sprache könnten unter anderem auf die umfangreichere englischsprachige Datenbasis während der Vortrainierungsphase zurückzuführen sein.

Zusammenfassend lässt sich festhalten, dass Transformer-Modelle ein hohes Potenzial für die Sentiment-Analyse englischer Texte besitzen, auch wenn sie ohne weiteres Training eingesetzt werden.

6.3 Gesamtvergleich: Qualität vs. Laufzeit

Die Ergebnisse der durchgeführten Experimente zeigen deutliche Unterschiede zwischen den betrachteten Modellklassen in Bezug auf die Vorhersagequalität und Rechenaufwand.

Während komplexere Modelle in der Regel höhere F1-Werte erzielen, hängt dies häufig mit deutlich längeren Ausführungszeiten zusammen.

Klassische Machine-Learning-Modelle liefern insgesamt stabile und mittlere F1-Werte bei vergleichsweise geringeren Laufzeiten. Insbesondere die Logistische Regression überzeugt durch sehr geringe Ausführungszeiten bei nur kleinem Qualitätsverlust. Die Support Vector Machine erzielte ähnliche Ergebnisse, ist aber aufgrund der höheren rechnerischen Komplexität deutlich zeitintensiver.

Lexikonbasierte Modelle zeichnen sich durch kürzere Laufzeiten aus, insbesondere bei englischsprachigen Texten, zeigten aber die niedrigsten F1-Scores. Da diese Ansätze ohne Lernphase auskommen und ausschließlich auf vordefinierten Wortlisten basieren, ist ihre Fähigkeit zur Berücksichtigung des Kontexts stark eingeschränkt.

Transformer-basierte Modelle wie BERT und RoBERTa erreichen sowohl für die deutsche als auch für die englische Sprache die höchsten F1-Scores. Gleichzeitig weisen sie im Vergleich zu den anderen Ansätzen erhöhte Laufzeiten auf, wegen der komplexen Modellarchitektur und den hohen Rechenbedarf. Der Einsatz spezieller Hardware trägt dazu bei, die Ausführungszeiten zu reduzieren, bleibt jedoch mit erhöhten technischen Anforderungen verbunden.

Zusammenfassend lässt sich festhalten, dass kein Modell in allen betrachteten Kriterien perfekt ist. Die Wahl eines geeigneten Ansatzes hängt vielmehr von den jeweiligen Anforderungen ab. Während Transformer-Modelle eine hohe Klassifikationsqualität liefern, bieten klassische Machine-Learning-Modelle einen guten Kompromiss zwischen Leistung und Rechenaufwand. Lexikonbasierte Verfahren eignen sich vor allem für einfache oder ressourcenschonende Szenarien.

7 Webanwendung

In diesem Kapitel wird die Webanwendung vorgestellt, die die Ergebnisse der Sentiment-Analyse visualisiert und interaktiv nutzbar macht.

7.1 Systemarchitektur

Da dieses Projekt kundenorientiert ausgelegt ist, wurde eine einfache Programmierschnittstelle (API) sowie ein Prototyp einer Webanwendung entwickelt. Ziel dieser Anwendung ist es, die Ergebnisse der Sentiment-Analyse verständlich darzustellen und zeitliche Trends sichtbar zu machen.

Für die Visualisierung der Analyseergebnisse wurde eine Weboberfläche realisiert, über die Nutzer gesammelte Kundenbewertungen hochladen können.(Abbildung 7.1) Nach der Verarbeitung der Daten werden die Ergebnisse angezeigt.

Abbildung 7.1: Webanwendung - Start

Die Grundstruktur der Webanwendung wurde mit HTML und CSS realisiert. Diese Technologien werden für die Strukturierung und Gestaltung von Websites verwendet. Für die Interaktivität, wie das Hochladen von Dateien oder die Auswahl von Analyseoptionen, wurde mithilfe von JavaScript umgesetzt.

Da es sich um einen Prototyp handelt, ist der Funktionsumfang bewusst reduziert:

- Es kann nur eine einzelne Datei hochgeladen werden. (Abbildung 7.2)
- Unterstützt wird das JSON-Format, da es eine einfache Weiterverarbeitung der Textdaten ermöglicht.
- Wird eine neue Datei ausgewählt, ersetzt sie automatisch die vorherige.
- Ohne ausgewählte Datei kann der Analyseprozess nicht gestartet werden; eine entsprechende Fehlermeldung wird angezeigt.

Abbildung 7.2: Webanwendung - Upload Data

Zusätzlich hat der Nutzer die Möglichkeit, den Zeitraum für die Analyse festzulegen:

- Auswahl vordefinierter Intervalle: jährlich, monatlich, wöchentlich, täglich (Abbildung 7.3)
- Alternativ: individueller Zeitraum über konkrete Datumsangaben (Abbildung 7.4)
- Wird kein Zeitraum definiert, wird der gesamte Datensatz berücksichtigt.

Abbildung 7.3: Webanwendung - Upload Data (Interval ausgewählt)

Abbildung 7.4: Webanwendung - Upload Data (Datum ausgewählt)

7.2 Ablauf der Sentiment Analyse

Die Webanwendung kommuniziert über eine einfache API mit dem Backend, in dem die eigentliche Sentiment-Analyse durchgeführt wird. Nach dem Hochladen der Daten werden

die Texte an die API übermittelt und verarbeitet. Der Ablauf der Analyse ist im Prototypen wie folgt automatisiert:

Zunächst wird die hochgeladene JSON-Datei in der Datenbank abgelegt und anschließend in einem Pandas-Dataframe umgewandelt. Falls ein bestimmter Zeitraum für die Analyse ausgewählt wurde, wird der Dataframe anhand der Zeitangaben entsprechend gefiltert.

Im nächsten Schritt erfolgt die eigentliche Sentiment-Analyse. Dabei wird nicht nur die Sentiment-Vorhersage für jeden Kommentar berechnet, sondern zusätzlich eine grobe Zuordnung möglicher Themen durchgeführt, auf die sich die Kommentare beziehen könnten.

Im Rahmen des Prototyps wird aktuell nur ein einzelnes Sprachmodell verwendet. Wie in Abschnitt 4.1 beschrieben, konzentriert sich das Projekt hauptsächlich auf den deutschsprachigen Raum. Basierend auf den Evaluationsergebnissen aus Kapitel 6 wurde RoBERTa als geeignetes Modell identifiziert, da es für die deutsche Sprache die beste Leistung gezeigt hat.

7.3 Visualisierung

Nach der erfolgreichen Durchführung der Sentiment-Analyse werden die Ergebnisse für den Nutzer visuell aufbereitet. Ziel der Visualisierung ist es, die Analyseergebnisse übersichtlich darzustellen und die Interpretation der Kundenbewertungen zu erleichtern. Für die Erstellung der Diagramme wird die Python-Bibliothek *matplotlib* verwendet. Diese Bibliothek bietet eine umfangreiche Auswahl an Funktionen zur Erstellung statistischer Diagramme.

Beim entwickelten Prototyp lag der Fokus auf einer klaren und nachvollziehbaren Darstellung der Ergebnisse und nicht auf interaktiven oder animierten Visualisierungen. *Matplotlib* ermöglicht eine präzise Kontrolle über Achsen, Beschriftungen und Layout und eignet sich daher besonders für reproduzierbare Auswertungen. Zudem lässt sich *matplotlib* in bestehende Python-basierte Umgebung integrieren, beispielsweise in Kombination mit *pandas*. Dadurch können Analyse und Visualisierung ohne zusätzliche Abhängigkeiten oder komplexe Frontend-Technologien umgesetzt werden. Für den vorgesehenen Anwendungsfall ist der Funktionsumfang von *matplotlib* daher vollständig ausreichend.

Für jede Produktart werden separate Plots erzeugt (Abbildungen 7.5, 7.6, 7.7). Jeder dieser Plots besteht aus drei zentralen Grafiken. Die erste Grafik zeigt die Verteilung der numerischen Kundenbewertungen, um einen allgemeinen Eindruck der Bewertungslage zu vermitteln. Die zweite Grafik stellt die vom Modell vorhergesagten Sentiment-Kategorien dar und unterscheidet dabei zwischen positiven, neutralen und negativen Kommentaren. Die dritte Grafik visualisiert die häufigsten Aspekte, die in negativen Bewertungen genannt werden. Dadurch erhält der Nutzer einen Überblick darüber, welche Eigenschaften eines Produkts besonders kritisch wahrgenommen werden.

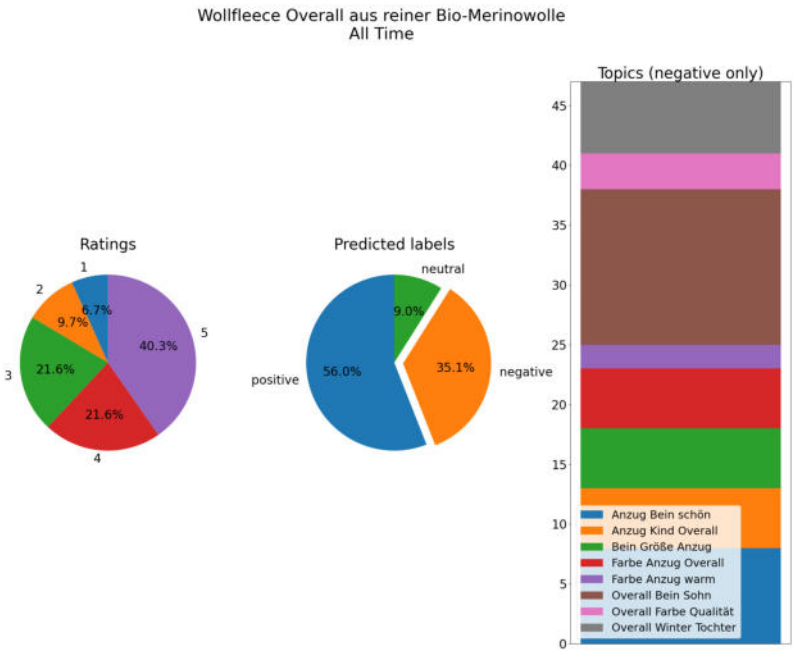

Abbildung 7.5: Webanwendung - Plot Beispiel Produkt 1

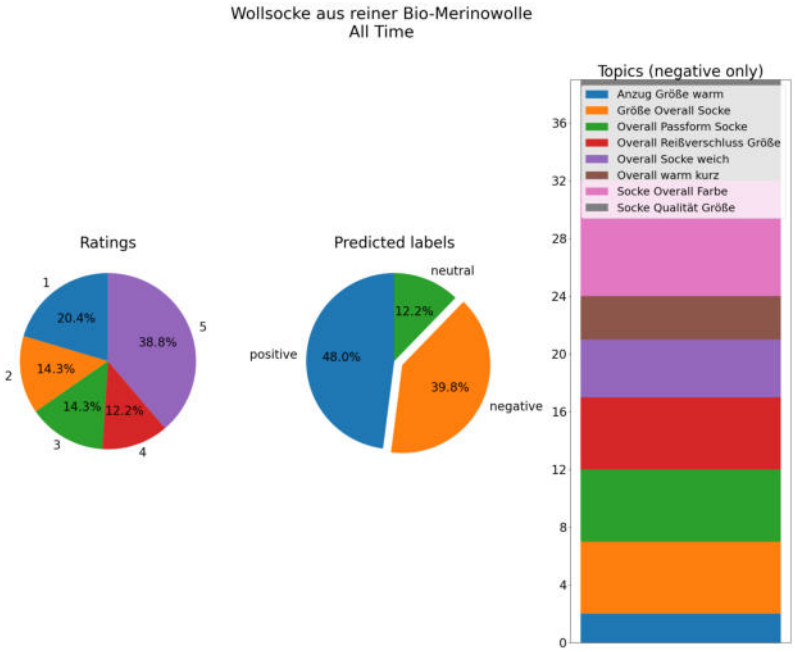

Abbildung 7.6: Webanwendung - Plot Beispiel Produkt 2

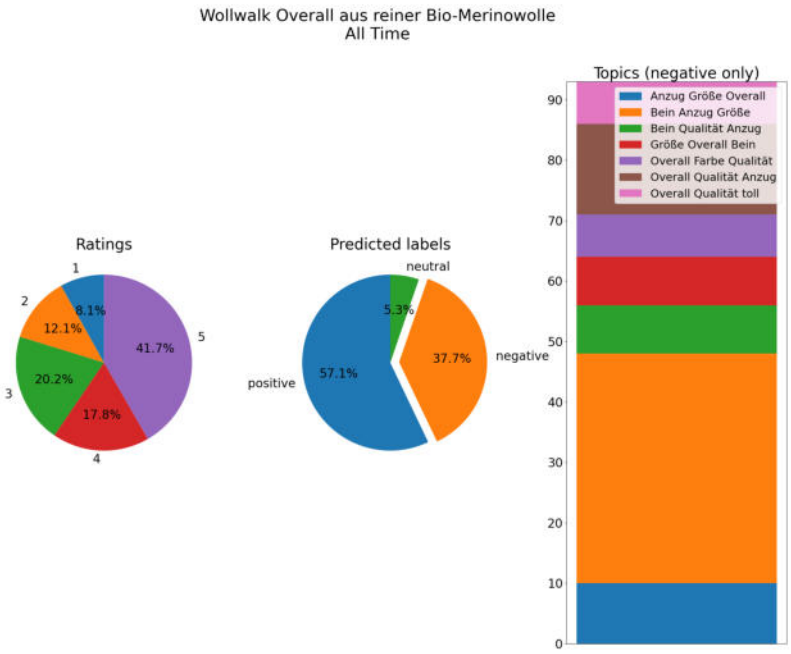

Abbildung 7.7: Webanwendung - Plot Beispiel Produkt 3

Da es sich bei der Webanwendung um einen Prototyp handelt, können einzelne automatisch ermittelte Aspekte teilweise ungewöhnlich oder wenig aussagekräftig erscheinen, beispielsweise Begriffe wie „Bein Anzug Größe“ oder „Farbe Anzug warm“. Diese Ergebnisse ergeben sich aus der vereinfachten Themenextraktion.

Die Bestimmung der Themen erfolgt mithilfe der Python-Bibliothek *gensim*, das dominante Begriffe innerhalb von Texten statistisch identifiziert. Dabei werden häufig vorkommende Wortkombinationen als potenzielle Themen interpretiert, ohne die semantische Bedeutung vollständig zu erfassen [ŘS10].

Ein geplanter weiterer Plot sollte den zeitlichen Verlauf der Verkaufszahlen in Verhältnis zu Kundenbewertungen zeigen. Ziel dieser Darstellung wäre es gewesen, mögliche Zusammenhänge zwischen negativen Bewertungen und Veränderungen der Verkaufszahlen aufzuzeigen. Dies konnte aufgrund fehlender Daten von „Firma X“ im Rahmen des Projekts jedoch nicht umgesetzt werden.

Insgesamt sollen die Visualisierungen dem Nutzer als Grundlage dienen, um Schwachstellen von Produkten zu identifizieren und gezielte Verbesserungsvorschläge abzuleiten.

8 Fazit und Ausblick

8.1 Zusammenfassung

Im Rahmen dieser Arbeit wurden verschiedene Ansätze der Sentiment-Analyse zur Auswertung von Kundenbewertungen untersucht und systematisch miteinander verglichen. Dabei wurden sowohl deutsch- als auch englischsprachige Texte betrachtet und die Modelle hinsichtlich ihrer Klassifikationsleistung bewertet. Zusätzlich wurde die praktische Umsetzbarkeit der Sentiment-Analyse in einem kundenorientierten Anwendungsszenario demonstriert.

Die Ergebnisse zeigen, dass sich die betrachteten Modellklassen deutlich in ihrer Leistungsfähigkeit unterscheiden. Lexikonbasierte Modelle weisen kurze Laufzeiten auf, erzielen jedoch insgesamt niedrige F1-Werte, da sie ohne Lernphase auskommen und den Kontext von Texten nur eingeschränkt erfassen können.

Klassische Machine-Learning-Modelle liefern stabile und vergleichbare Ergebnisse bei moderatem Rechenaufwand. Ihre Leistungsfähigkeit hängt stark von der Textvorverarbeitung und der gewählten Merkmalsrepräsentation ab. Bei geeigneter Vorverarbeitung stellen sie einen praktikablen Kompromiss zwischen Analysequalität und Effizienz dar.

Die höchsten Klassifikationsleistungen wurden von transformerbasierten Modellen erzielt. Diese profitieren von ihrer Fähigkeit, kontextuelle Informationen zu berücksichtigen, sind jedoch mit erhöhtem Rechenaufwand verbunden. Die Ergebnisse verdeutlichen, dass insbesondere die Größe und Passung der Trainingsdaten sowie ein gezieltes Fine-Tuning entscheidend für die Leistungsfähigkeit dieser Modelle sind.

Ergänzend wurde ein Prototyp einer Webanwendung umgesetzt, der das Hochladen von Kundenbewertungen, deren automatische Analyse sowie eine übersichtliche Visualisierung der Ergebnisse ermöglicht. Der Prototyp zeigt exemplarisch, wie Sentiment-Analyse in einem praxisnahen, kundenorientierten Kontext eingesetzt werden kann.

Insgesamt lässt sich festhalten, dass die Wahl eines geeigneten Sentiment-Analyse-Ansatzes stark von den jeweiligen Anforderungen abhängt. Die Arbeit zeigt, dass Sentiment-Analyse ein wirkungsvolles Instrument zur Auswertung von Kundenfeedback darstellt und wertvolle Erkenntnisse zur Unterstützung datenbasierter Entscheidungen liefern kann.

8.2 Ausblick

Für weiterführende Arbeiten bieten sich mehrere Erweiterungsmöglichkeiten an. So könnte die automatische Trenderkennung auf Basis zeitlich aggregierter Sentiment-Daten vertieft

untersucht werden, um langfristige Veränderungen in der Wahrnehmung von Produkten zu erfassen.

Die entwickelte Webanwendung kann durch zusätzliche Funktionen erweitert werden, beispielsweise durch eine Archivierung der erzeugten Visualisierungen, Filter- und Löschoptionen für hochgeladene Datensätze sowie die Möglichkeit, mehrere Dateien gleichzeitig zu verarbeiten. Auch die Unterstützung weiterer Sprachen wäre denkbar.

Methodisch könnte eine vertiefte Analyse rekurrenter neuronaler Netze (RNNs) zusätzliche Erkenntnisse liefern, insbesondere im Vergleich zu den in dieser Arbeit untersuchten Modellen.

Darüber hinaus konnte aufgrund fehlender Verkaufsdaten die potenzielle Abhängigkeit zwischen Kundenbewertungen und Verkaufszahlen nicht untersucht werden. Zukünftige Studien könnten diese Beziehung analysieren, sofern entsprechende Daten verfügbar sind, und so weitere Einblicke in den Einfluss von Kundenfeedback auf den Unternehmenserfolg gewinnen.

Die aufgezeigten Erweiterungen zeigen, dass die Ergebnisse dieser Arbeit eine solide Basis für weitere Forschung und praxisnahe Anwendungen im Bereich der Sentiment-Analyse darstellen.

Literaturverzeichnis

[Ali19] Thomas Lotze Alice Zheng Amanda Casari. *Merkmalskonstruktion für Machine learning : Prinzipien und Techniken der Datenaufbereitung / deutsche Übersetzung von Thomas Lotze*. 1.Auflage. Heidelberg : O'Reilly, 2019. ISBN: 978-3-96009-093-9.

[And24] Melanie Andersen. *Computerlinguistische Methoden für die Digital Humanities*. Dischingerweg 5, 72070 Tübingen: Narr Francke Attempto Verlag GmbH + Co.KG, 2024. ISBN: 978-3-8233-8579-0.

[Aur18] Kristian Rother Aurélien Géron. *Praxiseinstieg Machine Learning mit Scikit-Learn und TensorFlow : Konzepte, Tools und Techniken für intelligente Systeme / deutsche Übersetzung von Kristian Rother*. 1.Auflage. Heidelberg : O'Reilly, 2018. ISBN: 978-3-96009-061-8.

[Dev+19] Jacob Devlin u. a. „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Hrsg. von Jill Burstein, Christy Doran und Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, Juni 2019, S. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423/.

[Dud26] Duden. *Duden - Bewertung*. 2026. URL: https://www.duden.de/rechtschreibung/Bewertung (besucht am 26. 01. 2026).

[Guh+20] Oliver Guhr u. a. „Training a Broad-Coverage German Sentiment Classification Model for Dialog Systems". In: *Proceedings of The 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, Mai 2020, S. 1620–1625. URL: https://www.aclweb.org/anthology/2020.lrec-1.202.

[HG14] C. Hutto und Eric Gilbert. „VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text". In: *Proceedings of the International AAAI Conference on Web and Social Media* 8.1 (Mai 2014), S. 216–225. DOI: 10.1609/icwsm.v8i1.14550. URL: https://ojs.aaai.org/index.php/ICWSM/article/view/14550.

[Hut26] C.J. Hutto. *VADER-Sentiment-Analysis*. 2026. URL: https://github.com/cjhutto/vaderSentiment?tab=readme-ov-file#citation-information (besucht am 26. 01. 2026).

[IBM26a] IBM. *Stemming and Lemmatization*. 2026. URL: https://www.ibm.com/de-de/think/topics/stemming-lemmatization (besucht am 26. 01. 2026).

[IBM26b] IBM. *Was ist die Stimmungsanalyse?* 2026. URL: https://www.ibm.com/de-de/think/topics/sentiment-analysis (besucht am 26. 01. 2026).

[IBM26c] IBM. *Was ist NLP (Verarbeitung natürlicher Sprache)?* 2026. URL: https://www.ibm.com/de-de/think/topics/natural-language-processing (besucht am 26. 01. 2026).

[JM26] Daniel Jurafsky und James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*. 3rd. Online manuscript released January 6, 2026. 2026. URL: https://web.stanford.edu/~jurafsky/slp3/.

[Lor25] Steven Loria. *TextBlob: Simplified Text Processing*. 2025. URL: https://textblob.readthedocs.io/en/dev/ (besucht am 26. 01. 2026).

[NY03] Tetsuya Nasukawa und Jeonghee Yi. „Sentiment analysis: Capturing favorability using natural language processing“. In: Jan. 2003, S. 70–77. DOI: 10.1145/945645.945658.

[ŘS10] Radim Řehůřek und Petr Sojka. „Software Framework for Topic Modelling with Large Corpora“. English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. http://is.muni.cz/publication/884893/en. Valletta, Malta: ELRA, Mai 2010, S. 45–50.

[Sci26] Scikit-Learn. *train_test_split*. 2026. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html (besucht am 26. 01. 2026).

[Zhu+21] Liu Zhuang u. a. „A Robustly Optimized BERT Pre-training Approach with Post-training“. eng. In: *Proceedings of the 20th Chinese National Conference on Computational Linguistics*. Hrsg. von Sheng Li u. a. Huhhot, China: Chinese Information Processing Society of China, Aug. 2021, S. 1218–1227. URL: https://aclanthology.org/2021.ccl-1.108/.