FernUniversität in Hagen
Fakultät für Mathematik und Informatik

Artificial Intelligence Group

Stimmungsanalyse mit Twitter

Projektpraktikumsbericht
vorgelegt von
Anne Huber, Andreas Franke, Felix Lindner, Burak Özkan, Milomir Soknic

Zusammenfassung

Dieser Bericht beschreibt die Durchführung und Ergebnisse eines Projekts zur Stimmungsanalyse von Tweets. Ziel des Projekts war es, die Effektivität verschiedener maschineller Lernverfahren, einschließlich klassischer Methoden und *Deep-Learning*-Ansätze, bei der Klassifizierung der Stimmung von Tweets zu untersuchen. Dazu wurden verschiedene Vorverarbeitungsschritte und Vektorisierungsverfahren für klassische Methoden angewendet und evaluiert. Für die *Deep-Learning* Ansätze wurden *Finetuning*-Ansätze für *BERT*-basierte Modelle angewandt. Des Weiteren wurden für *DeepSeek*-R1 Modelle Prompt-basierte Ansätze mit und ohne Berücksichtigung von Aspekten untersucht. Die Ergebnisse zeigen, dass *Deep-Learning*-Modelle, insbesondere *BERT*-basierte Modelle mit *Finetuning* sowie aktuelle *LLMs*, eine höhere Genauigkeit bei der Stimmungsanalyse erzielen als klassische maschinelle Lernverfahren.

1. Einleitung

Twitter, heute unter dem Namen *X* bekannt, zählt zu den bedeutendsten sozialen Netzwerken, auf dem täglich Millionen von Nutzenden ihre Meinungen zu unterschiedlichen Themen äußern. Aufgrund der enormen Menge an Tweets bietet die Plattform wertvolle Einblicke in die öffentliche Meinung und gesellschaftliche Diskurse [20]. Die Stimmungsanalyse, ein Teilgebiet der Computerlinguistik, ermöglicht es, Emotionen und Meinungen automatisiert zu erkennen und zu kategorisieren. Insbesondere im Bereich der Web Science trägt diese Methode dazu bei, Dynamiken in soziotechnischen Systemen besser zu verstehen [3, 15].

Im Rahmen dieses Projekts wurden verschiedene Verfahren des maschinellen Lernens auf einen Twitter-Datensatz angewandt, um die mittels Tweets ausgedrückte Stimmung der Nutzer zu analysieren. Als Datengrundlage dient der *Sentiment140*-Datensatz der auf *HuggingFace* [6] verfügbar ist. Der Datensatz enthält 1,6 Mio. klassifizierte Tweets

und wurde von Go et al. erstellt [10]. Ziel der Analysen ist es, unterschiedliche Ansätze, einschließlich Methoden aus dem Bereich *Deep Learning* (DL), hinsichtlich ihrer Fähigkeit zur Stimmungsanalyse zu evaluieren.

In diesem Bericht wird zunächst die Aufgabenverteilung im Team sowie die genutzten Tools beschrieben. Anschließend werden die Schritte der Datenaufbereitung, die eingesetzten Modelle und die durchgeführten Experimente erläutert. Abschließend erfolgt eine Diskussion der Ergebnisse, gefolgt von einem Ausblick auf Optimierungsmöglichkeiten und weiterführende Forschungsansätze.

2. Aufgabenverteilung

Die Themen Literaturrecherche und Ideenfindung, Überprüfung der Datenqualität des gegebenen Datensatzes, Suche nach einem neuen Datensatz und die explorative Datenanalyse des *Sentiment140*-Datensatzes wurden von allen Projektmitgliedern bearbeitet. Darüber hinaus wurde von allen Projektmitgliedern ein klassisches Verfahren angewandt und unterschiedliche Kombinationen von Vorverarbeitungsschritten in Bezug auf die Genauigkeit getestet. Außerdem wurden die Zwischen- und Abschlusspräsentation nach der initialen Erstellung gemeinsam überarbeitet. Die Abschnitte des Abschlussberichts wurden anfänglich von einzelnen Mitgliedern geschrieben und anschließend gemeinsam überarbeitet. Im Folgenden sind die Aufgaben aufgezählt, die nicht von allen Mitgliedern durchgeführt wurden.

2.1. Anne Huber

- Kommunikation mit den Projektbetreuern
- Protokollerstellung der Treffen
- Halten der Zwischenpräsentation
- Klassische Verfahren: *k-Nearest Neighbors* (KNN)
- Abschlussbericht: 2, 3, 7

2.2. Andreas Franke

- Klassische Verfahren: SVM
- Implementierung Framework für die klassischen Verfahren
- Implementierung *Finetuning*-Logik für die BERT-basierten Ansätze
- Implementierung Prompting-Logik für die *DeepSeek*-Modelle
- Durchführung Komplett-Abläufe für klassische Verfahren
- Durchführung Abläufe für die BERT-basierten Ansätze
- Durchführung *Prompting*-Ansätze (*DeepSeek* R1 1.5B, 8B, 32B)
- Abschlussbericht: 4, 5.1, 6.1.1, 6.2, 6.3.1

2.3. Felix Lindner

- Klassische Verfahren: Entscheidungsbäume
- Erweiterung des Frameworks für die klassischen Verfahren
- Implementierung *Finetuning*-Logik für die BERT-basierten Ansätze
- Durchführung Abläufe für die BERT-basierten Ansätze
- Durchführung *Prompting*-Ansatz (*DeepSeek* R1 70B)
- Halten der Abschlusspräsentation
- Abschlussbericht: 5.2, 6.1.2, 6.3.2

2.4. Burak Özkan

- Klassische Verfahren: Naiver Bayes Klassifikator
- Erstellung der Abschlusspräsentation
- Abschlussbericht: 8

2.5. Milomir Soknic

- Klassische Verfahren: Logistische Regression
- Erstellung der Zwischenpräsentation
- Erstellung der Abschlusspräsentation
- Abschlussbericht: 1

3. Teaminterne Organisation

Zur asynchronen Kommunikation der Projektmitglieder wurde eine private Lerngruppe in einer Discord-Instanz der Studenten der FernUniversität Hagen eingerichtet. Dort wurden auch die regelmäßigen Treffen über die vorhandenen Sprachkanäle durchgeführt. Die Intervalle der Treffen wurden in Absprache mit allen Mitgliedern und in Abhängigkeit von den durchgeführten Schritten festgelegt. In den Treffen wurden die erreichten Ergebnisse besprochen sowie mögliche nächste Schritte diskutiert und geplant.

Als Programmiersprache wurde *Python* gewählt, da bei allen Projektmitgliedern unter anderem durch die vorherige Teilnahme am Modul *Einführung in Maschinelles Lernen* diesbezüglich Vorkenntnisse vorhanden waren und viele Bibliotheken für Maschinelles Lernen (ML) und DL in *Python* verfügbar sind. Zur Verwaltung der verwendeten Python-Pakete und zur Vereinheitlichung und Prüfung der erstellten Dateien wurden *Poetry* [21] sowie *pre-commit* [22] verwendet.

Für die klassischen ML-Verfahren wurde insbesondere die Bibliothek *scikit-learn* eingesetzt. Darüber hinaus wurde für die Anwendung der DL-Ansätze und der eigenen Ansätze die *transformers*-Bibliothek von *Hugging Face* genutzt.

Zur Verwaltung der Projektdateien und der gemeinsamen Arbeit wurde ein *GitHub Repository* [7] eingerichtet. Für neue Ansätze und Ergebnisse sowie Änderungen an Code und Dokumenten wurden in *GitHub Branches* angelegt und über *Pull-Requests* überprüft und diskutiert.

Die Abfolge der einzelnen Projektschritte orientierte sich an dem *Data Science Life Cycle* [27, Abb. 2]. Dieser wurde während der Anwendung der klassischen Methoden einmal durchlaufen. Dies beinhaltet die wiederholte Durchführung von Schritten, wie *Obtain Data* und *Data Exploration* oder die parallele Ausführung von *Data Preparation* und *Model Estimation* bei der Durchführung der unterschiedlichen klassischen Verfahren. Im Anschluss, bei der Anwendung der DL-Modelle, wurde gleichermaßen mehrfach über gewisse Schritte des *Data Science Life Cycle* iteriert.

4. Datensätze und Problemstellung

4.1. Datensätze

Es wurden die separaten Trainings- und Testdatensätze des von Go et al. [10] erstellten *Sentiment140*-Datensatzes verwendet.

4.1.1. Trainingsdatensatz

Der verwendete Datensatz *Sentiment140* enthält 1,6 Mio. modifizierte Tweets. Die Tweets wurden im Zeitraum von April 2009 bis Juni 2009 erstellt. Jedem Tweet ist eine binäre Stimmungsklasse (positiv/negativ) zugeordnet.

Die Tweets wurden mithilfe der Twitter-API gesammelt, indem nach Tweets gesucht wurde, die bestimmte Emoticons mit positiver oder negativer Bedeutung enthalten (siehe Tabelle 3 im Artikel zum Datensatz [10, S. 4]). Anhand der verwendeten Emoticons wurden die Tweets in Klassen mit positiver und negativer Stimmung eingeteilt. Tweets, die sowohl positive als auch negative Emoticons enthalten, sind nicht im Datensatz enthalten. Go et al. [10] weisen darauf hin, dass die Stimmungsklassen nicht fehlerfrei sind ([10, Abschnitt 2.2]) und es sich um *Noisy Labels* handelt. Als *Noisy Labels* werden Klassenzuordnungen bezeichnet, die nicht fehlerfrei sind und die Qualität eines Datensatzes beeinträchtigen können [26]. Für beide Stimmungsklassen enthält der Datensatz jeweils 800.000 Einträge.

Die Tweet-Texte im Datensatz wurden so angepasst, dass die Emoticons, die zur Einteilung verwendet wurden, entfernt wurden. Weiterhin enthält der Datensatz keine Retweets und keine Duplikate.

4.1.2. Testdatensatz

Von Go et al. [10] wurde ein Testdatensatz mit 498 Tweets erstellt, der 177 Tweets mit positiver Stimmung und 182 Tweets mit negativer Stimmung enthält. Die restlichen Tweets sind als neutral klassifiziert und wurden bei der Evaluierung der Modelle nicht berücksichtigt. Die Tweets wurden über die Twitter-API durch Anfragen mit spezifischen *Query*-Ausdrücken ausgewählt und manuell mit Stimmungsklassen versehen. Die *Query*-Ausdrücke sind in Tabelle 4 des Artikels von Go et al. [10, S. 5] aufgeführt und sind im Testdatensatz enthalten. Die Tweets enthielten nicht in jedem Fall Emoticons.

4.2. Problemstellung

Im Rahmen der Stimmungsanalyse für Twitter wird versucht, die Tweets in zwei oder mehr Klassen einzuteilen [30].

Die Analyse der Stimmung von Tweets wird allerdings als besonders herausfordernd angesehen [1, 9, 30]. Giachanou und Crestani [9] nennen neben der Längenbeschränkung von Tweets auf 140 Zeichen (bzw. 280 seit 2017) insbesondere die informelle Art von Tweets als Herausforderung in Bezug auf Stimmungsanalysen. Agarwal et al. [1] und Zimbra et al. [30] weisen darauf hin, dass aufgrund der Längenbeschränkung besonders häufig Abkürzungen, Emoticons und andere Zeichen mit spezieller Bedeutung oder Umgangssprache in Tweets verwendet werden.

Das Ziel dieser Arbeit ist es, bestehende Methoden zur automatischen Stimmungsanalyse von Tweets anzuwenden und zu evaluieren. Dabei sollen sowohl klassische ML-Verfahren als auch moderne Deep-Learning-Ansätze untersucht werden.

Die zentrale Problemstellung lautet: Wie effektiv sind verschiedene ML-Verfahren bei der Stimmungsanalyse von Tweets?

Insbesondere soll untersucht werden, welche Ansätze die besten Ergebnisse in Bezug auf die Genauigkeit der Klassifikation liefern. Darüber hinaus sollen die Herausforderungen und Limitationen der Stimmungsanalyse von Tweets identifiziert und diskutiert werden.

5. Ansätze

In diesem Abschnitt werden die Ansätze zur Stimmungsanalyse von Tweets beschrieben, die wir verwendet haben. In den Unterabschnitten 5.1 und 5.2 werden die verwendeten klassischen Ansätze und DL-basierten Ansätze aufgezählt und beschrieben.

5.1. Klassische Ansätze

Als klassische Ansätze zur Stimmungsanalyse von Tweets wurden folgende überwachte Lernverfahren ausgewählt, da diese besonders häufig in der Literatur zu finden sind [18, 29, 30]:

- **Logistische Regression (LR)** — *Scikit-learn*-Modell *LogisticRegression*
- ***Support Vector Machine* (SVM)** — *Scikit-learn*-Modell *LinearSVC*
- **Naive Bayes (NB) Klassifikator** — *Scikit-learn*-Modell *BernoulliNB*

Die Verfahren KNN und Entscheidungsbäume wurden, mit dem Wissen, dass diese vermutlich nicht für die Stimmungsanalyse geeignet sind, ebenfalls überprüft, aber verworfen, da diese auf dem von uns verwendeten Datensatz schlechtere Ergebnisse erzielten und darüber hinaus längere Laufzeiten benötigten.

Eine detaillierte, mathematische Einführung in die genannten klassischen Verfahren ist zu finden in [11, vgl. die Kapitel 4.4 für LR, 4.5 und 12 für SVM, 6.6.3 für den NB-Klassifikator, 13.3 für KNN und 9.2 für Entscheidungsbäume].

5.2. Deep Learning Ansätze

Nach Wankhade et al. [29] haben in den letzten Jahren *Transformer*-basierte Modelle *Long Short-Term Memory* (LSTM) und *Convolutional Neural Network* (CNN)-Modelle in der Stimmungsanalyse abgelöst. Aus diesem Grund haben wir uns für die Verwendung von *Transformer*-basierten *Large Language Model*s (LLMs) entschieden.

Es wurden *Bidirectional Encoder Representations from Transformers* (BERT)-basierte und *DeepSeek*-basierte Modelle als Vertreter der *Transformer*-Modelle ausgewählt.

BERT-basierte Modelle wurden ausgewählt, da diese in der Literatur als besonders erfolgreich in der Stimmungsanalyse beschrieben werden [5]. Die BERT-basierten Modelle und Ansätze sind in Unterabschnitt 5.2.1 beschrieben.

Die *DeepSeek*-R1-basierten Modelle wurden im Rahmen des eigenen Ansatzes aufgrund ihrer Aktualität und der allgemein guten Ergebnisse für unsere Problemstellung evaluiert. Die Modelle und Ansätze sind in Unterabschnitt 5.2.2 beschrieben.

Transformer basieren auf der von Vaswani et al. beschriebenen Architektur *Multi-layer Bidirectional Transformer Encoder* [28].

***Transformer*-Architektur** Die Architektur eines *Multi-layer Bidirectional Transformer Encoder* besteht aus einem *Encoder*, der eine Eingabe auf eine interne Repräsentation abbildet, und einem *Decoder*, der aus dieser internen Repräsentation eine Ausgabesequenz generiert. In jedem Schritt verwendet der *Decoder* seine Ausgabe aus dem vorherigen Schritt.

Der *Encoder* besteht aus N gleichen Schichten, die jeweils aus einem *Multi-head Self-Attention*-Mechanismus und einem voll-vernetzten neuronalen Netz bestehen. Der *Decoder* besteht ebenfalls aus N gleichen Schichten, die sich zusammensetzen aus einer Schicht *Multi-head Self-Attention*, einem voll-vernetzten neuronalen Netz und zusätzlich einer Schicht *Multi-head Self-Attention*, welche die Ausgabe des *Encoders* verarbeitet.

Training von *Transformern* Das Training von *Transformer*-Modellen besteht aus zwei Schritten, dem *Pre-Training* und dem *Finetuning* [23]. Beim *Pre-Training* wird das Modell unüberwacht mit großen Datenmengen trainiert. Im zweiten Schritt, dem *Finetuning*, wird das Modell mit überwachten Lernverfahren oder *Reinforcement Learning* [4, 5] auf den jeweiligen Anwendungsfall angepasst.

5.2.1. *BERT*-basierte Ansätze

BERT ist eine *Transformer*-basierte LLM-Variante, die von Devlin et al. entwickelt wurde [5]. Devlin et al. haben gezeigt, dass die BERT-Modelle sehr gute Ergebnisse in der Stimmungsanalyse erzielen und durch *Finetuning* mit vergleichsweise wenig Daten und Trainingszeit auf spezifische Anwendungen angepasst werden können. Aufbauend auf BERT wurde von Liu et al. [16] das Modell *Robustly optimized BERT approach* (RoBERTa) entwickelt, das durch Optimierungen im *Pre-Training* und *Finetuning* für viele Aufgaben bessere Ergebnisse erzielt.

Von Barbieri et al. [2] wurde gezeigt, dass für Twitter-spezifische Klassifikationsprobleme RoBERTa-basierte Modelle durch *Finetuning* auf Twitter-Daten bessere Ergebnisse erzielen. Aus diesem Grund haben wir uns für die Verwendung des von Barbieri et al. entwickelten Modells entschieden, welches bereits mittels *Finetuning* an Twitter-Daten angepasst wurde.

Das Modell wurde zum einen ohne weitere Anpassungen an dem Testdatensatz evaluiert und zum anderen nach einer Anpassung mittels *Finetuning* an den Trainingsdatensatz ebenfalls an dem Testdatensatz ausgewertet.

Zum Vergleich haben wir das von BERT destillierte Modell *DistilBert* [24] verwendet. Bei der Modelldestillation wird ein kleines Modell darauf trainiert, das Verhalten eines großen Modells zu replizieren [24]. Auf diesem Modell wurde ebenfalls ein *Finetuning* auf dem Trainingsdatensatz und eine Evaluation auf dem Testdatensatz durchgeführt.

5.2.2. *DeepSeek*-basierte Ansätze

Als eigenen Ansatz haben wir verschiedene vom *DeepSeek*-R1-Modell destillierte Modelle [4] verwendet. Es wurde versucht, diese in einem ersten Ansatz mit *Finetuning* auf unseren Datensatz anzupassen. Aufgrund der hohen Anforderungen an die Hardware und der langen Trainingszeiten haben wir uns für einen anderen Ansatz entschieden. Das *Finetuning* konnten wir lediglich für das kleinste destillierte Modell ausführen.

In einem zweiten Ansatz haben wir die *DeepSeek*-R1-Modelle mittels *Ollama* [19] ausgeführt und mittels Prompt-Anfragen eine Klassifizierung der Tweets durch die Modelle durchgeführt. Die Prompt-Anfragen wurden dabei mit und ohne Verwendung des *Query*-Ausdrucks des Testdatensatzes ausgeführt.

6. Experimente

6.1. Experimentaufbau

Für die Experimente wurden unterschiedliche Schritte in Abhängigkeit der Verfahren durchgeführt.

6.1.1. Klassische Ansätze

Für die klassischen Verfahren wurde neben der Genauigkeit der Klassifikation für unterschiedliche Modelle auch analysiert, inwiefern Vorverarbeitungsschritte und Tokenisierungsverfahren die Genauigkeit der Klassifikation beeinflussen.

Datenvorverarbeitung Im Rahmen der Datenvorverarbeitung wurden die verwendeten Stoppwörter und Normalisierungsverfahren variiert.

Nach [17, S.27] werden unter dem Begriff Stoppwörter Wörter verstanden, die einen geringen Informationsgehalt haben und deshalb aus Texten entfernt werden, wie beispielsweise *und* oder *oder*. Es wurden drei verschiedene Verfahren zur Behandlung von Stoppwörtern verwendet: Beibehaltung aller Stoppwörter, Verwendung der Standard-*Natural Language Toolkit* (NLTK) Stoppwortliste und Verwendung einer eigenen Stoppwortliste zur Entfernung spezifischer Stoppwörter.

Normalisierungsverfahren dienen dazu, die Worte oder Token in Texten zu vereinheitlichen [17, S.28]. Es wurden drei verschiedene Verfahren zur Normalisierung der Worte verwendet: keine Normalisierung, Lemmatisierung (mit *WordNet Lemmatizer*) und Stemming (mit *Porter Stemmer*).

Das Training der Modelle wurde auf Basis der vorverarbeiteten Daten durchgeführt. Im Rahmen der Vorverarbeitung der Daten wurden die Tweets bereinigt, die einzelnen Token normalisiert und abschließend die Stoppwörter entfernt. In Abschnitt A.1 im Appendix ist der Algorithmus detaillierter beschrieben (s. Algorithmus 1) und ein Beispiel gegeben.

Training und Evaluation Auf Basis der vorverarbeiteten Daten wurden die Modelle trainiert und evaluiert.

Die Texte der vorverarbeiteten Tweets werden mittels Vektorisierungsverfahren in numerische Repräsentationen bzw. Vektoren transformiert. Es wurden zwei Vektorisierungsverfahren verwendet: *Term Frequency-Inverse Document Frequency* (TF-IDF)-Vektorisierung und *Hash*-basierte Vektorisierung[1]. TF-IDF Vektorisierung ist ein Verfahren zur Gewichtung von Termen in Texten, das die Häufigkeit der Terme in einem Dokument und die inverse Häufigkeit von Dokumenten mit diesem Termen berücksichtigt [17, S. 119]. Die *Hash*-basierte Vektorisierung ist ein Verfahren, das die Wörter in einem Dokument mittels einer Hash-Funktion in numerische Werte umwandelt, um die Wörter in einen Vektor fester Länge zu kodieren [25].

Für die verwendeten Vektorisierungsverfahren wurden unterschiedliche Konfigurationen von n-Grammen verwendet. n-Gramme bezeichnen nach [14, S.33], eine Sequenz von n aufeinanderfolgenden Wörtern. Für die Vektorisierungsverfahren wurden Instanzen mit Kombinationen von 1-Grammen, 2-Grammen und 3-Grammen verwendet.

Während des Trainings werden die Modelle auf den Trainingsdaten trainiert und auf den Validierungsdaten evaluiert. Die Genauigkeit der Modelle wird abschließend auf den Testdaten evaluiert. Algorithmus 2 beschreibt die Schritte für das Training und die Evaluation der Modelle.

6.1.2. Deep Learning Ansätze

Für die DL Ansätze, bei denen BERT-basierte Modelle verwendet wurden, wurden die Standard-Tokenizer der *Hugging Face* Modelle verwendet, so dass keine weiteren Vorverarbeitungsschritte oder Vektorisierungsverfahren durchgeführt wurden.

***Finetuning* der BERT-Modelle** Die BERT-basierten Modelle *twitter-roberta-base-sentiment* und *distilbert-base-uncased* wurden über die *Transfomers* Python-Bibliothek von *Hugging Face* zunächst mit Hilfe des Trainingsdatensatzes trainiert und ausgeführt.

Für das *Finetuning* wurde die Standardkonfiguration der Bibliothek verwendet. Variiert wurden die Datensatzgröße, also die Anzahl der Tweets, die für das *Finetuning* verwendet wurden, und die Lernrate. Die Werte für die Lernrate lagen zwischen 10^{-4} und 10^{-6} und die Datensatzgröße zwischen 2500 und 20000. Die Werte sind in Tabelle 4 aufgeführt und orientieren sich an dem Vorgehen von Barbieri et al. [2].

Verwendung der *DeepSeek*-Modelle Für den eigenen Ansatz haben wir zuerst versucht, die von *DeepSeek*-R1 destillierten Modelle durch *Finetuning* auf den Datensatz zu trainieren. Dafür wurde den Modellen jeweils zwei zusätzliche voll-vernetzte Schichten angefügt. Diese Architektur entspricht dem Aufbau des *twitter-roberta-base-sentiment* Modells zur Klassifikation.

Die erste voll-vernetzte Schicht hatte als Ein- und Ausgabe die Dimensionen der letzten Schicht der Modelle. Die zweite voll-vernetzte Schicht hatte als Eingabe die Ausgabe der ersten voll-vernetzten Schicht und als Ausgabe die Anzahl der Klassen.

[1] Für beide Verfahren wurden die Implementierungen der *scikit-learn* Bibliothek verwendet.

Modell	Normalisierung	Stoppwortliste	Anz. Merkmale	n-Gramme	Genauigkeit
LR	Porter	eig. Liste	250.000	(1,2)	0.859
SVM	Porter	-	maximal	(1,3)	0.858
SVM	Porter	eig. Liste	50.000	(1,2)	0.858
LR	WordNet	-	maximal	(1,3)	0.858
SVM	WordNet	-	maximal	(1,3)	0.858

Tabelle 1: Top 5 Modelle nach Testgenauigkeit angeordnet (Mittelwerte von drei Ausführungen). Die Bezeichner $(1, k)$ in der Spalte *n-Gramme* geben an, dass k-Gramme mit $k \in \{1, \cdots, 3\}$ verwendet wurden.

Die Durchführung des *Finetunings* war für das kleinste destillierte *DeepSeek*-R1-Modell mit 1,5 Milliarden Parametern noch möglich. Ab dem nächstgrößerem Modell (*DeepSeek-R1:7B*) wurden die Hardware-Anforderungen zu groß[2].

Als weiteren Ansatz verwendeten wir die DeepSeek-R1-Modelle nur in der Ausführung und ließen die Stimmung per Prompt klassifizieren. Die Modelle wurden mit *Ollama* ausgeführt und per Pythonskript angefragt. Die Anfragen wurden mit und ohne *Query*-Ausdruck (siehe Abschnitt 4.1.2) durchgeführt.

Die Anfragen mit Query-Ausdruck hatten folgende Struktur (wobei die Platzhalter $QueryTerm$ und $Tweet$ durch die entsprechenden Werte ersetzt wurden):

> Tweet sentiment? Sentiment Topic: $QueryTerm$
> Answer with positive or negative. Provide reasoning in JSON.
> Tweet: $Tweet$

6.2. Modell-Parameter und Evaluationsmetriken

Für die ausgewählten klassischen Modelle wurden die Standard-Parameterwerte von *scikit-learn* verwendet. Für die BERT-basierten Modelle und die *DeepSeek*-Modelle wurden die veröffentlichten Modelle verwendet bzw. auf diesen im Rahmen des *Finetunings* aufgesetzt.

Nach [29] werden für die Evaluierung von Klassifikationsmodellen vor allem das *Genauigkeitsmaß*, die *Präzision* oder das *F1-Maß* verwendet. Die Klassenverteilung für die positive und negative Klasse der Trainingsdaten ist ausgeglichen und die Verteilung der Testdaten ist ebenfalls relativ ausgeglichen. Aufgrund der einfachen Interpretierbarkeit wurde deshalb das *Genauigkeitsmaß* als Evaluationsmetrik verwendet.

6.3. Ergebnisse

Eine Übersicht über die maximalen Genauigkeiten je Modell und Ansatz ist im Anhang in Diagramm 1 dargestellt.

[2]Vergleiche mit *Stochastic Gradient Descent* (SGD) Optimierung: 7 Mrd. Parameter × 2 Byte je Gewicht × 2 Byte je Gradient = 26 GB

	Testgenauigkeit über alle Parameter-Kombinationen				
Modell	Mittelwert	Median	Std.-Abweichung	Minimum	Maximum
LR	0.819	0.822	0.022	0.727	0.861
SVM	0.809	0.813	0.024	0.730	0.858
NB	0.778	0.784	0.049	0.685	0.852

Tabelle 2: Statistiken der Testgenauigkeit für die Modelle LR, SVM und NB.

6.3.1. Klassische Ansätze

In Tabelle 1 sind die Top 5 Modelle nach Testgenauigkeit sortiert aufgeführt. Die maximal erzielte Genauigkeit beträgt $0,859$.

Sensitivität Modell In Tabelle 2 sind die Statistiken der Testgenauigkeit für die Modelle SVM, LR und NB über alle Parameter-Kombinationen aufgeführt.

Die LR-Modelle erzielen im Mittel die höchsten Genauigkeiten über alle Parameter-Kombinationen. Weiterhin ist die Standardabweichung der Genauigkeiten für die LR mit $0,022$ am niedrigsten.

Modelle auf Basis der SVM erzielen im Mittel und im Median 1% niedrigere Genauigkeiten. Mit NB-Modelle wurden im Mittel die niedrigsten Genauigkeiten erreicht, wobei die Standardabweichung am höchsten war.

Sensitivität Umgang mit Stoppwörtern Über alle Parameter-Kombinationen hinweg ist die Genauigkeit im Mittel für die Datensätze mit Entfernung der Stoppwörter höher als für die Datensätze ohne Entfernung. Weiterhin ist die Genauigkeit mit der eigens definierten Stoppwortliste höher als mit der NLTK-Liste.

Sensitivität Normalisierungsverfahren Für die unterschiedlichen Normalisierungsverfahren ergeben sich keine signifikanten Unterschiede in der Genauigkeit der Modelle.

Sensitivität Vektorisierungsverfahren und n-Gramme Für alle Modelle ist die Genauigkeit höher, wenn das TF-IDF-Vektorisierungsverfahren verwendet wird.

Für alle Modelle steigt die Genauigkeit mit der Anzahl an berücksichtigten Merkmalen und, wenn mehr als nur Unigramme berücksichtigt werden. Die Verwendung von Uni- und Bigrammen führt im Mittel zu den höchsten Genauigkeiten.

6.3.2. *Deep Learning* Ansätze

***Finetuning* der BERT-Modelle** Mit dem auf aktuelleren Twitter-Daten trainierten Modell *twitter-roberta-base-sentiment* wurden Genauigkeiten von $0,83$ auf dem Testdatensatz erzielt. Durch *Finetuning* wurden Genauigkeiten von $0,922$ für das Modell *twitter-roberta-base-sentiment* und $0,849$ für das Modell *distilbert-base-uncased* erreicht.

Modell	Genauigkeit mit Query-Ausdruck	Genauigkeit ohne Query-Ausdruck
DeepSeek-r1-70B	0.977	0.930
DeepSeek-r1-32B	0.966	0.927
DeepSeek-r1-8B	0.955	0.916
DeepSeek-r1-1.5B	0.883	0.824

Tabelle 3: Genauigkeit bei Verwendung der *DeepSeek*-Modelle ohne *Finetuning*

Kleinere Lernraten führen zu höheren Genauigkeiten für das Modell *twitter-roberta-base-sentiment*, während für das Modell *distilbert-base-uncased* höhere Lernraten zu besseren Ergebnissen führen. Dies ist vermutlich darauf zurückzuführen, dass das Modell *twitter-roberta-base-sentiment* bereits auf Twitter-Daten trainiert wurde und das *distilbert-base-uncased* Modell lediglich auf einem allgemeinen Korpus.

Verwendung der *DeepSeek*-Modelle Das *Finetuning* des kleinsten *DeepSeek*-Modells lieferte Genauigkeiten von bis zu $0,866$.

Für die Verwendung der *DeepSeek*-Modelle ohne *Finetuning* mittels direkter Anfragen wurden Genauigkeiten bis zu $0,977$ erzielt. Hier gilt, dass die Genauigkeit steigt je mehr Parameter das destillierte Modell hat. Weiterhin ist die Genauigkeit höher, wenn die Anfragen mit einem *Query*-Term durchgeführt werden. Die Ergebnisse sind in Tabelle 3 zusammengefasst.

7. Ausblick

Es gibt mehrere Ansätze, um die Vorgehensweise und die Ergebnisse der Stimmungsanalyse zu optimieren.

7.1. *Noisy Label*

Wie in Abschnitt 4.1.1 beschrieben, führt die Art, wie der Trainingsdatensatz *Sentiment140* erstellt wurde, zu *Noisy Label*. Dies kann bei der Verwendung einfacherer Modelle zu schlechteren Klassifikationsergebnissen führen sowie die Anzahl der benötigten Trainingsbeispiele oder die Komplexität der Modelle erhöhen [8], wie man auch an den moderaten Ergebnissen der verwendeten klassischen Verfahren ($\approx 85\%$) und den Resultaten von Go et al. ($79 - 83\%$) sieht (vgl. Tabelle 6 in [10]).

Um mit *Noisy Labels* umzugehen, können robustere Modelle oder Methoden der Datenbereinigung verwendet werden, also Datensätze zu entfernen oder sogenannte Semi-überwachte Lernverfahren anzuwenden. Alternativ können auch *Label Noise* tolerante Lernverfahren angewandt werden (vgl. Abschnitt 3 in [8]).

Eine grundsätzlich andere Herangehensweise ist die Verwendung eines Datensatzes mit höherer Datenqualität, um *Noisy Labels* zu vermeiden.

7.2. *Aspect Based Sentiment Analysis*

Die erlangten Ergebnisse der unterschiedichen Modelle können durch die Verwendung zusätzlicher Informationen verbessert werden, wie die Ergebnisse der Experimente mit den *DeepSeek*-basierten Modellen zeigen.

Bei der *Aspect Based Sentiment Analysis* werden explizite oder implizite Begriffe (*aspects*) aus dem Tweet extrahiert und die Stimmung des Tweets bezüglich dieser Begriffe klassifiziert [13].

Ein Beispiel aus [13] zeigt, dass der Satz „*The restaurant was expensive, but the menu was great.*“ bezüglich dem expliziten Begriff *menu* positiv und bezüglich dem impliziten Begriff *price* negativ klassifiziert wird.

7.3. Große *Deep Learning* Modelle

Die Feinabstimmung großer *DeepSeek* Modelle mit dem *Sentiment140* Datensatz stieß auf Rechenkapazitätsgrenzen. Während das Modell *DeepSeek-R1-1.5B* noch abgestimmt werden konnte, waren die Speicheranforderung während des Trainings für die größeren Modelle mit 8, 32 oder 70 Mrd. Parametern zu hoch für die zur Verfügung stehenden Rechenressourcen.

Mehr Rechenkapazitäten oder Methoden wie die *Low Rank Adaptation* (LoRA) [12], bei der nur eine kleine Anzahl neuer Parameter zur Feinabstimmung genutzt wird, können hier Abhilfe schaffen.

8. Zusammenfassung und Fazit

In diesem Projekt wurden verschiedene Methoden des maschinellen Lernens zur Stimmungsanalyse auf dem Sentiment140-Twitter-Datensatz evaluiert. Drei klassische Algorithmen – LR, SVM und NB-Klassifikator – erreichten maximale Genauigkeiten von bis zu 85%. Zwei Transformer-Modelle (*distilbert-base-uncased* und *twitter-roberta-base-sentiment*) wurden durch *Finetuning* angepasst und erzielten Genauigkeiten von 85% bzw. 92%. Zusätzlich wurde versucht, die durch Modelldestillation erzeugten *DeepSeek-R1*-Modelle mittels *Finetuning* anzupassen. Aufgrund der benötigten Hardware-Ressourcen war dies lediglich für die kleinste Variante (*Deepseek-R1-1.5B*) möglich, die Genauigkeiten von bis zu 87% lieferte. Das größte verwendete *DeepSeek-R1* basierte Modell erreichte die höchste Genauigkeit mit 93% im *Zero-Shot*-Ansatz. Mit einem aspektbasierten Ansatz wurden mit den *DeepSeek-R1* basierten Modellen Genauigkeiten bis zu 98% erzielt.

Die Ergebnisse zeigen, dass domänenspezifische Transformer-Modelle bei der Stimmungsanalyse höhere Genauigkeiten erzielen als die analysierten klassischen ML-Verfahren. Obwohl die LLM mit den meisten Parametern die höchsten Genauigkeiten erreichten, kann in der Praxis der Einsatz vortrainierter kleinerer Transformer-Modelle (z. B. *RoBERTa*) für Twitter-Stimmungsanalysen vorteilhafter sein – insbesondere bei begrenzten Hardware-Ressourcen oder dem Einsatz in Anwendungen mit strengen Latenzanforderungen.

Literatur

[1] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca J Passonneau. Sentiment analysis of twitter data. In *Proceedings of the workshop on language in social media (LSM 2011)*, pages 30–38, 2011.

[2] Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online, November 2020. Association for Computational Linguistics.

[3] Tim Berners-Lee, Wendy Hall, James Hendler, Nigel Shadbolt, and Daniel J. Weitzner. Web science: Understanding the emergence of macro-level features on the world wide web. *Communications of the ACM*, 51(7):60–69, 2006.

[4] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie,

Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[5] 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.

[6] Hugging Face. sentiment140 dataset. `https://huggingface.co/datasets/stanfordnlp/sentiment140`, 2023. aufgerufen am 22.03.2025.

[7] Andreas Franke, Anne Huber, Burak Özkan, and Milomir Soknic. Github repository. `https://github.com/felLind/propra_webscience_ws24/`, 2025. aufgerufen am 22.03.2025.

[8] Benoît Frénay and Ata Kabán. A comprehensive introduction to label noise. In *The European Symposium on Artificial Neural Networks*, 2014.

[9] Anastasia Giachanou and Fabio Crestani. Like it or not: A survey of twitter sentiment analysis methods. *ACM Computing Surveys (CSUR)*, 49(2):1–41, 2016.

[10] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009, 2009.

[11] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York, New York, NY, 2009.

[12] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[13] Yan Cathy Hua, Paul Denny, Jörg Wicker, and Katerina Taskova. A systematic review of aspect-based sentiment analysis: domains, methods, and trends. *Artificial Intelligence Review*, 57(11), September 2024.

[14] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd edition, 2025. Online manuscript released January 12, 2025.

[15] Bing Liu. *Sentiment analysis and opinion mining*, volume 5 of *Synthesis lectures on human language technologies*. Morgan & Claypool Publishers, 2012.

[16] 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.

[17] Christopher D Manning. *An introduction to information retrieval*. 2009.

[18] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113, 2014.

[19] Meta. Ollama. `https://ollama.com/`, 2025. aufgerufen am 22.03.2025.

[20] Alexander Pak, Patrick Paroubek, et al. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, pages 1320–1326, 2010.

[21] Poetry. Poetry. `https://python-poetry.org/0`, 2025. aufgerufen am 22.03.2025.

[22] Pre-Commit. Pre-commit. `https://pre-commit.com/`, 2025. aufgerufen am 22.03.2025.

[23] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[24] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.

[25] scikit-learn developers. sckit learn feature extraction. `https://scikit-learn.org/stable/api/sklearn.feature_extraction.html`, 2025. aufgerufen am 22.03.2025.

[26] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE transactions on neural networks and learning systems*, 34(11):8135–8153, 2022.

[27] Victoria Stodden. The data science life cycle: a disciplined approach to advancing data science as a science. *Commun. ACM*, 63(7):58–66, June 2020.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[29] Mayur Wankhade, Annavarapu Chandra Sekhara Rao, and Chaitanya Kulkarni. A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review*, 55(7):5731–5780, 2022.

[30] David Zimbra, Ahmed Abbasi, Daniel Zeng, and Hsinchun Chen. The state-of-the-art in twitter sentiment analysis: A review and benchmark evaluation. *ACM Transactions on Management Information Systems (TMIS)*, 9(2):1–29, 2018.

Erklärung

Ich erkläre, dass ich die schriftliche Ausarbeitung zum Fachpraktikum selbstständig und ohne unzulässige Inanspruchnahme Dritter verfasst habe. Ich habe dabei nur die angegebenen Quellen und Hilfsmittel verwendet und die aus diesen wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht. Die Versicherung selbstständiger Arbeit gilt auch für enthaltene Zeichnungen, Skizzen oder graphische Darstellungen. Die Ausarbeitung wurde bisher in gleicher oder ähnlicher Form weder derselben noch einer anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Mit der Abgabe der elektronischen Fassung der endgültigen Version der Ausarbeitung nehme ich zur Kenntnis, dass diese mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiate geprüft werden kann und ausschließlich für Prüfungszwecke gespeichert wird.

.....
(Ort, Datum) (Unterschrift)

.....
(Ort, Datum) (Unterschrift)

.....
(Ort, Datum) (Unterschrift)

.....
(Ort, Datum) (Unterschrift)

.....
(Ort, Datum) (Unterschrift)

A. Algorithmen

A.1. Datenaufbereitung

Hinweis: Die Datensätze enthalten weitere Daten je Tweet, wie beispielsweise die Sentiment Klasse. Der Umgang mit diesen Daten wurde hier aus Gründen der Übersichtlichkeit nicht aufgeführt.

Für die unterschiedlichen Parameter-Ausprägungen für die Normalisierungsverfahren und Stoppwortlisten wurden separate Datensätze erstellt.

A.1.1. Datenaufbereitung - Algorithmus

Die Datensätze wurden gemäß des folgenden Algorithmus erzeugt.

Algorithm 1 Datenaufbereitung

```
1:  procedure DatasetPreparation(dataset, normalizerFunction, stoppwords)
2:      function SanitizeTweet(text)
3:          text ← Entferne URLs, Nutzer, Hashtags und Sonderzeichen aus text
4:          return text
5:      end function
6:      function NormalizeTweet(text, normalizerFunction, stoppwords)
7:          tokens ← Zerlege text mit TweetTokenizer in Token
8:          for i ← 1 to |tokens| do
9:              tokens[i] ← normalizerFunction(tokens[i])
10:         end for
11:         tokens ← Entferne Stoppwörter aus tokens gemäß Stoppwortliste stoppwords
12:         text ← Füge Elemente aus tokens zu einem Text zusammen
13:         return text
14:     end function
15:     for i ← 1 to |dataset| do
16:         dataset[i] ← SanitizeTweet(dataset[i])
17:         dataset[i] ← NormalizeTweet(dataset[i], normalizerFunction, stoppwords)
18:     end for
19:     save dataset
20: end procedure
```

A.1.2. Datenaufbereitung - Beispiel

Im folgenden ist ein Beispiel für die Datenaufbereitung eines Tweets dargestellt.

- **Original-Tweet**: *„@user I love this movie! http://example.com“*
- **Bereinigung**: *„I love this movie“*
- **Tokenisierung**: *„I“, „love“, „this“, „movie“*
- **Normalisierung**:

  - *Lemmatisierung*: *„I“, „love“, „this“, „movie“*
  - *Stemming*: *„I“, „lov“, „thi“, „movi“*
- **Stoppwörter Behandlung**: Ohne Stoppwörter „I“ und „this“: *„love“, „movie“*
- **Aufbereiteter Text**: *„love movie“*

A.2. Training und Evaluation der Modelle

Die Schritte für das Training und die Evaluation der Modelle sind, aufbauend auf dem durch Algorithmus 1 erzeugten Datensätzen wie folgt:

Algorithm 2 Training und Evaluation der Modelle

procedure ModelTraining($dataset, vectorizer, model$)
    $X, y \leftarrow$ Extrahiere Texte und Labels aus $dataset$
    $X \leftarrow$ Transformiere Texte in X in Vektoren mit Vektorizer $vectorizer$
    $X_{train}, X_{test}, y_{train}, y_{test} \leftarrow$ Teile X und y in Trainings- und Testdaten
    $model \leftarrow$ Trainiere $model$ auf X_{train} und y_{train}
    $accuracy \leftarrow$ Evaluiere $model$ auf X_{test} und y_{test}
    return $accuracy$
end procedure

B. Tabellen

Die folgende Tabelle zeigt die einzelnen Werte für Datensatzgröße und Lernrate, welche für das *Finetuning* der DL-Ansätze verwendet wurden.

Parameter	Werte
Datensatzgröße	2500, 5000, 7500, 10000, 15000, 20000
Lernrate	10^{-4}, $5 \cdot 10^{-5}$, 10^{-5}, $5 \cdot 10^{-6}$, 10^{-6}

Tabelle 4: Parameter für das *Finetuning* der BERT-Modelle

C. Diagramme

Im folgenden Diagramm sind für alle Modelle und Verfahren die maximalen Genauigkeiten dargestellt.

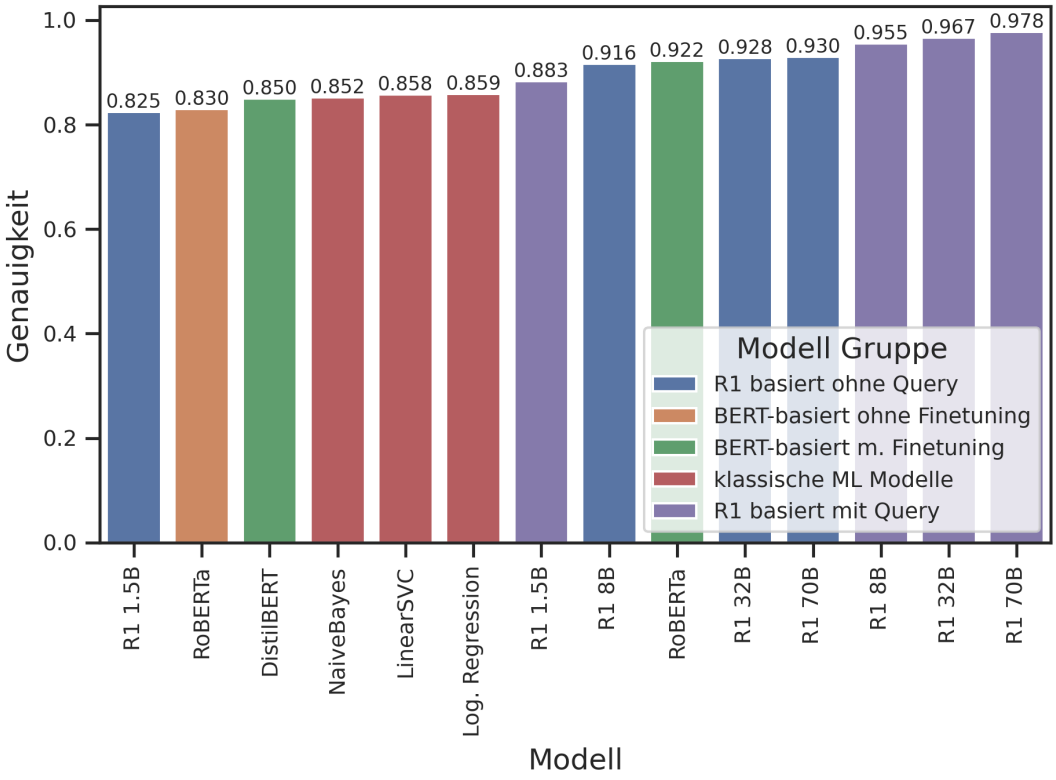

Abbildung 1: Maximale Genauigkeit der Modelle und Verfahren