Modellvergleich und Optimierung von NLP-Verfahren zur Klassifikation kryptobezogener Reddit-Beiträge

Digital Business University of Applied Sciences

Data Science & Business Analytics
DMI01 Data Mining
Daniel Ambach

Eingereicht von Dennis Reimer
Martrikelnummer: 190288
Datum: 16.04.2025

Zusammenfassung

Diese Arbeit befasst sich mit dem Vergleich und der Optimierung von NLP-Modellen zur Sentimentanalyse von Reddit-Beiträgen im Kontext von Kryptowährungen. Dabei werden sowohl klassische, lexikonbasierte Verfahren als auch moderne, transformerbasierte Sprachmodelle untersucht. Ziel ist es, diese Modelle im Hinblick auf ihre Fähigkeit zu bewerten, Aussagen auf Reddit zuverlässig in die Sentimentklassen bullish, neutral oder bearish einzuordnen. Besonderes Augenmerk liegt auf der Differenzierung zwischen vollständigen Reddit-Posts und kurzen Kommentaren, da sich diese in Struktur, Sprache und Tonalität deutlich unterscheiden.

Die Untersuchung basiert auf einem kombinierten Datensatz aus rund 2.000 GPT-gestützt gelabelten Reddit-Posts sowie einem öffentlich verfügbaren Kommentar-Datensatz aus einer IEEE-Studie, in dem das Sentiment direkt von den Beitragenden angegeben wurde. Die methodische Umsetzung folgt dem CRISP-DM-Prozessmodell, das eine strukturierte Herangehensweise an datengetriebene Projekte bietet. Zum Einsatz kamen unter anderem FinBERT und CryptoBERT als spezialisierte transformerbasierte Modelle, während VADER und TextBlob als klassische Vergleichsverfahren dienten.

Die Ergebnisse zeigen, dass CryptoBERT nach gezieltem Fine-Tuning auf Kommentaren eine sehr hohe Klassifikationsgüte erreichte (macro-F1: 0.94), während FinBERT bei den längeren Posts mit einem macro-F1 von 0.89 ebenfalls stark abschnitt. Beide Modelle konnten ihre Ausgangsleistung deutlich steigern. Die Arbeit unterstreicht damit die Relevanz domänenspezifischer Modellierung, ausreichender Datenbasis und sorgfältiger Hyperparameteroptimierung für den erfolgreichen Einsatz von Sentimentanalysen im Bereich von Social Media und Kryptowährungen.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Die Analyse von Meinungen und Emotionen im digitalen Raum hat in den letzten Jahren stark an Bedeutung gewonnen, besonders im Umfeld sozialer Medien und Finanzmärkte. Plattformen wie Reddit fungieren dabei als wichtige Diskussionsorte, an denen sich private und institutionelle Anlegerinnen und Anleger über Kryptowährungen, Markttrends und Anlagestrategien austauschen. Die automatisierte Auswertung solcher Inhalte mit Hilfe von Verfahren der Sentimentanalyse kann wertvolle Hinweise auf kollektive Stimmungen liefern und dadurch zu fundierteren Entscheidungen beitragen. Eine besondere Herausforderung bei Reddit besteht in der Vielfalt und Unstrukturiertheit der Texte. Die Inhalte reichen von ausführlichen Diskussionen bis hin zu kurzen, sarkastischen Kommentaren oder humorvollen Memes. Klassische, lexikonbasierte Ansätze stoßen hier schnell an ihre Grenzen, da ihnen das notwendige Kontextverständnis fehlt. Transformerbasierte Sprachmodelle wie BERT, FinBERT oder CryptoBERT gehen über eine reine Wortebene hinaus und berücksichtigen die Bedeutung eines Satzes im Gesamtzusammenhang. Gerade bei komplexen oder indirekt formulierten Aussagen ist dies ein entscheidender Vorteil (Huyen, 2022, Kapitel 4–5). Diese Arbeit verfolgt das Ziel, verschiedene NLP-Modelle zur Sentimentanalyse von Reddit-Beiträgen mit Bezug zu Kryptowährungen zu vergleichen und im nächsten Schritt gezielt zu optimieren. Untersucht werden sowohl transformerbasierte Modelle wie FinBERT, CryptoBERT, RoBERTa und DeBERTa als auch klassische Ansätze wie VADER und TextBlob. Die Modelle werden dabei auf zwei unterschiedlichen Textarten getestet: auf vollständigen Reddit-Posts und auf kurzen Kommentaren. Anschließend werden die leistungsfähigsten Modelle weiter angepasst und bewertet. Die methodische Umsetzung folgt dem CRISP-DM-Prozessmodell (Cross Industry Standard Process for Data Mining), das sich als Standard zur Strukturierung datengetriebener Projekte etabliert hat (Bramer, 2007, S. 15–16). Alle sechs Phasen, von der Zieldefinition bis zur abschließenden Bewertung, werden im Rahmen dieser Arbeit dokumentiert und umgesetzt. Für die Analyse wurden zwei Datensätze herangezogen: Zum einen eine Sammlung von etwa 2.000 Reddit-Posts, die mithilfe von GPT-basierten Verfahren gelabelt wurden, zum anderen ein frei zugänglicher Kommentar-Datensatz aus einer IEEE-Studie, bei dem das Sentiment direkt von den Verfassenden angegeben wurde (IEEE, 2023). Die technische Umsetzung erfolgte in Python unter Verwendung der Bibliotheken transformers, datasets und scikit-learn. Das Fine-Tuning der Modelle wurde auf GPU-beschleunigten Systemen durchgeführt (Downey & Lang, 2024, Kapitel 2).

Die Arbeit ist in fünf Kapitel gegliedert. Kapitel 2 stellt die theoretischen Grundlagen vor, insbesondere zu Sentimentanalyse, Modellarten und Metriken. Kapitel 3 beschreibt das methodische Vorgehen entlang des CRISP-DM-Prozesses. Kapitel 4 enthält die empirischen Ergebnisse und deren Auswertung. Kapitel 5 fasst die zentralen Erkenntnisse zusammen und gibt einen Ausblick auf weiterführende Fragestellungen.

2 Theoretischer Hintergrund

Um die eingesetzten Modelle und Bewertungsmethoden einordnen zu können, werden in diesem Kapitel die Grundlagen der Sentimentanalyse, relevante Modellarten und Bewertungsmetriken sowie das zugrunde liegende methodische Vorgehen vorgestellt.

2.1 Grundlagen der Sentimentanalyse

Sentimentanalyse ist ein Teilbereich des Natural Language Processing (NLP) und befasst sich mit der automatisierten Erkennung und Klassifikation von Meinungsäußerungen in Texten. Sie zielt darauf ab, Inhalte etwa als *positiv*, *negativ* oder *neutral* zu bewerten. Ihre Anwendungen reichen von Produktbewertungen über politische Stimmungsanalysen bis hin zur Finanzmarktforschung. Besonders in sozialen Medien wie Reddit stellt die Analyse eine Herausforderung dar: Die Texte sind oft kurz, informell, enthalten Ironie, Emojis oder Bezugnahmen auf vorherige Kontexte. Klassische regelbasierte Verfahren scheitern häufig an diesen Phänomenen, da ihnen das nötige Kontextverständnis fehlt. Deshalb kommen zunehmend tiefe Sprachmodelle auf Basis von Transformer-Architekturen zum Einsatz, die semantische Zusammenhänge modellieren können (Huyen, 2022, Kapitel 4).

2.2 Klassifikationsansätze im Vergleich

Grundsätzlich lassen sich Sentimentklassifikatoren in zwei Gruppen unterteilen: lexikonbasierte Verfahren und transformerbasierte Modelle. Die Lexikonbasierte Verfahren beinhalten Modelle wie VADER (Valence Aware Dictionary and sEntiment Reasoner) und TextBlob und verwenden vorab definierte Wortlisten, in denen Begriffen Polarisierungswerte zugewiesen sind. Die Gesamtwertung eines Textes ergibt sich durch Aggregation dieser Einzelwerte. VADER wurde speziell für Social Media konzipiert und berücksichtigt auch Großschreibung, Interpunktion und Emoticons. TextBlob verwendet ein einfaches regelbasiertes Verfahren in Kombination mit statistischen Textmerkmalen. Der Vorteil dieser Methoden liegt in ihrer schnellen Einsatzfähigkeit und hohen Interpretierbarkeit. Ihre Schwäche besteht darin, dass sie keine semantischen Zusammenhänge erkennen – ein wesentlicher Nachteil bei komplexer Sprache oder impliziten Meinungen.

Transformer sind eine moderne Architektur innerhalb künstlicher neuronaler Netze. Sie basieren auf dem Self-Attention-Mechanismus, der es ermöglicht, die Bedeutung eines Wortes im Kontext aller anderen Wörter zu erfassen. Dadurch sind sie besonders leistungsfähig bei der Analyse semantisch komplexer Texte (Huyen, 2022, Kapitel 4).
Besonders relevant für diese Arbeit sind:
FinBERT: Ein auf Finanzsprache spezialisiertes Modell, das auf Wirtschafts- und Börsentexten trainiert wurde.
CryptoBERT: Eine Variante, die auf Krypto-Diskussionen in Foren und sozialen Medien feinjustiert wurde.
RoBERTa und DeBERTa: Architekturverbesserungen von BERT, die auf allgemeinen Textdaten trainiert wurden und ohne Domänenspezialisierung arbeiten. Transformerbasierte Modelle sind rechenintensiv und benötigen größere Datenmengen, bieten jedoch ein deutlich tieferes Sprachverständnis – ein entscheidender Vorteil für Reddit-Daten mit ihrer hohen stilistischen Vielfalt.

2.3 Data Mining und CRISP-DM-Prozess

Die in dieser Arbeit durchgeführten Schritte lassen sich vollständig in das CRISP-DM-Modell (Cross Industry Standard Process for Data Mining) einordnen – ein weit verbreitetes Framework zur Strukturierung datengetriebener Projekte (Bramer, 2007, S. 15–16). Es besteht aus sechs klar definierten Phasen:

1. Business Understanding: Problemdefinition, Zielsetzung
2. Data Understanding: Beschreibung, Exploration und Bewertung der Datenqualität
3. Data Preparation: Bereinigung, Vorverarbeitung, Labeling und Formatierung
4. Modeling: Auswahl geeigneter Modelle und Konfigurationsparameter
5. Evaluation: Modellvergleich, Interpretation der Metriken
6. Deployment: Anwendung oder Transfer der Ergebnisse (in dieser Arbeit nur theoretisch)

Der Vorteil von CRISP-DM liegt in der Trennung von Analyse und Umsetzung, wodurch sowohl technische Details als auch fachliche Zielsetzungen klar nachvollziehbar bleiben (Bramer, 2007, S. 15-16). In dieser Arbeit bildet CRISP-DM die methodische Klammer, an der sich alle Kapitel orientieren.

2.4 Bewertung von Klassifikationsmodellen

Die Bewertung der Modellleistung erfolgt typischerweise über folgende Metriken:
Accuracy: Anteil korrekt klassifizierter Instanzen

F1-Score (macro): Harmonie von Precision und Recall über alle Klassen hinweg, sinnvoll bei unbalancierten Daten

Cohen's Kappa: Maß für die Übereinstimmung zwischen Vorhersage und Ground Truth, bereinigt um Zufallstreffer Besonders bei unausgeglichenen Datensätzen – wie bei Reddit-Kommentaren – ist der F1-Score (macro) die zuverlässigere Metrik. Er bewertet die Leistung jeder Klasse gleich und lässt sich daher gut für Modellvergleiche einsetzen (Huyen, 2022, Kapitel 10).

3 Implementierung und Methodik

Im Folgenden wird das methodische Vorgehen der Arbeit detailliert beschrieben. Die Struktur orientiert sich vollständig am CRISP-DM-Prozessmodell (Bramer, 2007, S. 15–16), das eine systematische Umsetzung datengetriebener Projekte ermöglicht.

3.1 Datenaufbereitung und Split

Nach dem Import der Datensätze wurden diese bereinigt, vereinheitlicht und in numerische Labels für die drei Sentimentklassen *bullish* (0), *neutral* (1) und *bearish* (2) überführt. Die Daten wurden anschließend in Trainings-, Validierungs- und Testdaten unterteilt. Dabei wurde explizit darauf geachtet, dass die Klassenverteilung auch in den Teilmengen erhalten bleibt (stratifizierter Split). Die konkrete Aufteilung und das Verhältnis (z. B. 60 % Training, 20 % Validierung, 20 % Test) variierten je nach Datensatz leicht und wurden direkt im Notebook durchgeführt. Auf einen externen Split-Helfer wurde bewusst verzichtet, um die Kontrolle über die Aufteilung direkt im Hauptnotebook zu behalten.

3.2 Baseline- Vergleich der Modelle

Zur Evaluation wurden zunächst sechs verschiedene Modelle auf einem festen Testset verglichen. Die Modellklasse reichte von lexikonbasierten Verfahren (VADER, TextBlob) bis zu transformerbasierten Klassifikatoren (FinBERT, CryptoBERT, RoBERTa, DeBERTa). Die lexikonbasierten Modelle erhielten eine explizite Schwellenwertlogik zur Klassenzuweisung:

```
score = model_obj.polarity_scores(text)["compound"]
preds.append(0 if score >= 0.05 else 2 if score <= -0.05 else 1)
```

Abbildung 1: Klassifizierung eines Textes in bullish, neutral und bearish basierend auf dem compound-Score von Vader

Nach der Inferenz wurde für jedes Modell ein Klassifikationsreport generiert. Die F1-Scores der drei Klassen wurden extrahiert und zentral gespeichert:

```python
def extract_f1_scores(classification_reports):
    f1_data = {}
    for model, report in classification_reports.items():
        f1_data[model] = {
            "bullish": report["bullish"]["f1-score"],
            "neutral": report["neutral"]["f1-score"],
            "bearish": report["bearish"]["f1-score"]
        }
    return f1_data
```

Abbildung 2: Funktion zur Extraktion der F1-Scores pro Klasse aus einem gespeicherten Klassifikationsreport

Die aufbereiteten Scores wurden später zur Erstellung der Balkendiagramme in Kapitel 4 verwendet.

3.3 Modelloptimierung mit Custom- Trainer

Für die zwei besten Modelle – FinBERT (Posts) und CryptoBERT (Kommentare) – wurde ein gezieltes Fine-Tuning durchgeführt. Dabei kam ein Custom-Trainer mit gewichteter Verlustfunktion zum Einsatz, um die unausgeglichene Klassenverteilung zu berücksichtigen:

```python
class WeightedLossTrainer(Trainer):
    def __init__(self, *args, loss_fn=None, **kwargs):
        super().__init__(*args, **kwargs)
        self.loss_fn = loss_fn

    def compute_loss(self, model, inputs, return_outputs=False, **kwargs):
        labels = inputs.pop("labels")
        outputs = model(**inputs)
        logits = outputs.logits
        loss = self.loss_fn(logits, labels)
        return (loss, outputs) if return_outputs else loss
```

Abbildung 3: Implementierung eines gewichteten Custom- Trainers zur Berücksichtigung unbalacierter Klassen

Es wurde mit unterschiedlichen Gewichtungen experimentiert, bis das beste Ergebnis gespeichert wurde. Die technische Umsetzung erfolgte mithilfe der Trainer-API von Hugging Face und der transformers-Bibliothek. Die gewählte Architektur erlaubte es, benutzerdefinierte Verlustfunktionen (z. B. CrossEntropyLoss mit Gewichtung) direkt in das Trainingssetup zu integrieren (Huyen, 2022, Kapitel 6).

3.4 Vergleich verschiedener Lernrraten

Ein zentrales Element der Optimierung war ein systematischer Vergleich unterschiedlicher Lernraten (2e-5 bis 1e-6). Für jede Rate wurde ein vollständiger Trainingslauf durchgeführt und die F1-Makro-Scores pro Epoche mitgeloggt.Die Trainingsargumente wurden so gesetzt, dass nach jeder Epoche eine Validierung erfolgte. Zusätzlich wurde ein Early-Stopping-Mechanismus integriert, um Überanpassung zu vermeiden.

```python
results = {}
lrs = [2e-5, 1e-5, 5e-6, 3e-6, 1e-6]

for lr in lrs:
    print(f"Starte Training für learning_rate = {lr}")

    output_dir = os.path.join(MODEL_PATHS["finbert_posts"], f"finetuned_lr_{lr}")
    logging_dir = os.path.join("../../logs/finbert_posts", f"lr_{lr}")
    os.makedirs(output_dir, exist_ok=True)
    os.makedirs(logging_dir, exist_ok=True)

    training_args = TrainingArguments(
        output_dir=output_dir,
        evaluation_strategy="epoch",
        save_strategy="epoch",
        learning_rate=lr,
        per_device_train_batch_size=8,
        per_device_eval_batch_size=8,
        num_train_epochs=10,
        weight_decay=0.01,
        load_best_model_at_end=True,
        metric_for_best_model="f1",
        logging_dir=logging_dir,
        logging_strategy="epoch",
        report_to="none",
        remove_unused_columns=False,
        disable_tqdm=True
    )
```

Abbildung 4:Trainingsschleife zum Vergleich verschiedener Lernraten mittels Trainingarguments in Hugging Face

Die daraus entstandenen Lernkurven sind in Kapitel 4 abgebildet und wurden zur Auswahl des besten Modells verwendet. Die Sensitivität der Modelle gegenüber Lernraten war besonders bei kleinen Datensätzen deutlich zu beobachten, was ein bekanntes Phänomen bei modernen NLP-Architekturen ist (Huyen, 2022, Kapitel 7).

4 Ergebnisse und Diskussion

Ziel dieses Abschnitts ist die systematische Analyse und Bewertung der getesteten Modelle, dabei wird zwischen den beiden Datentypen- Posts und Kommentare unterschieden. Zusätzlich werden die Effekte des Fine-Tunings auf die Modellperformance untersucht.

4.1 Modellvergleich auf untrainierten Baselines

Abbildung 5 und 6 zeigen die F1-Scores der getesteten Modelle auf dem ursprünglichen, untrainierten Stand. Dabei wurde jeweils ein gemeinsames Testset verwendet. Auffällig ist, dass kein Modell alle drei Klassen konsistent gut erkennt. Bei Posts (Abb. 5) erzielten klassische Methoden wie VADER und TextBlob relativ ausgewogene Ergebnisse für bullish und neutral, zeigten jedoch Schwächen bei bearish. Transformer-Modelle wie DeBERTa und RoBERTa lieferten bei bearish sehr niedrige F1-Scores (teils < 0.1), während CryptoBERT und FinBERT leicht bessere Ergebnisse in einzelnen Klassen erreichten – jedoch noch ohne

konsistente Stärke über alle Klassen hinweg.

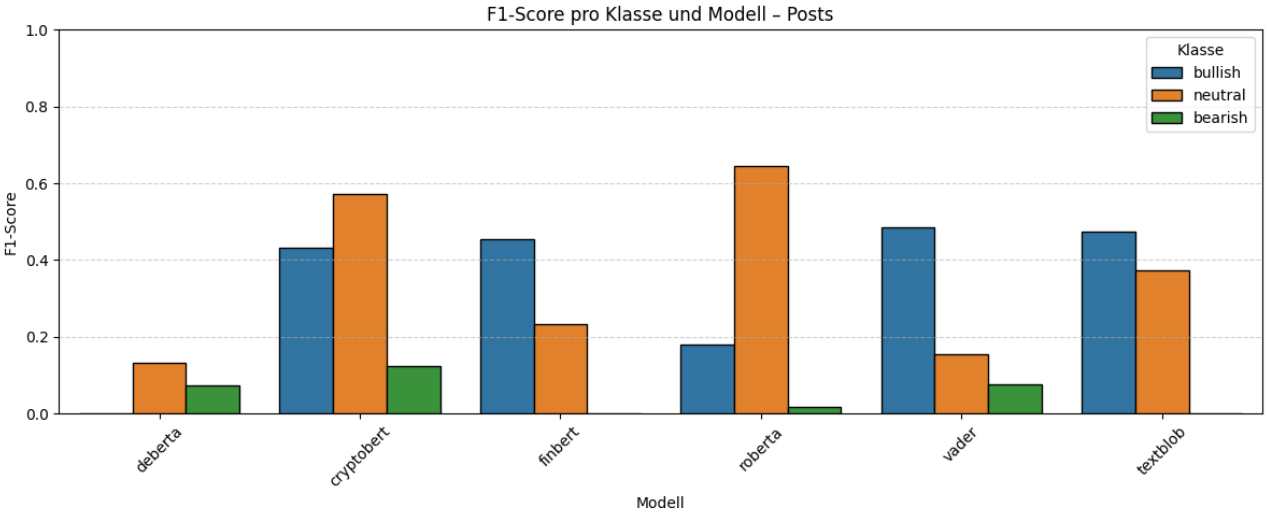

Abbildung 5: F1-Scores der Ausgsangsmodelle auf Reddit-Posts

Bei Kommentaren (Abb. 6) zeigte CryptoBERT auch ohne Fine-Tuning bereits eine sehr starke Performance (F1 bis zu 0.91 für *bullish*), während DeBERTa hier erneut nahezu vollständig versagte.

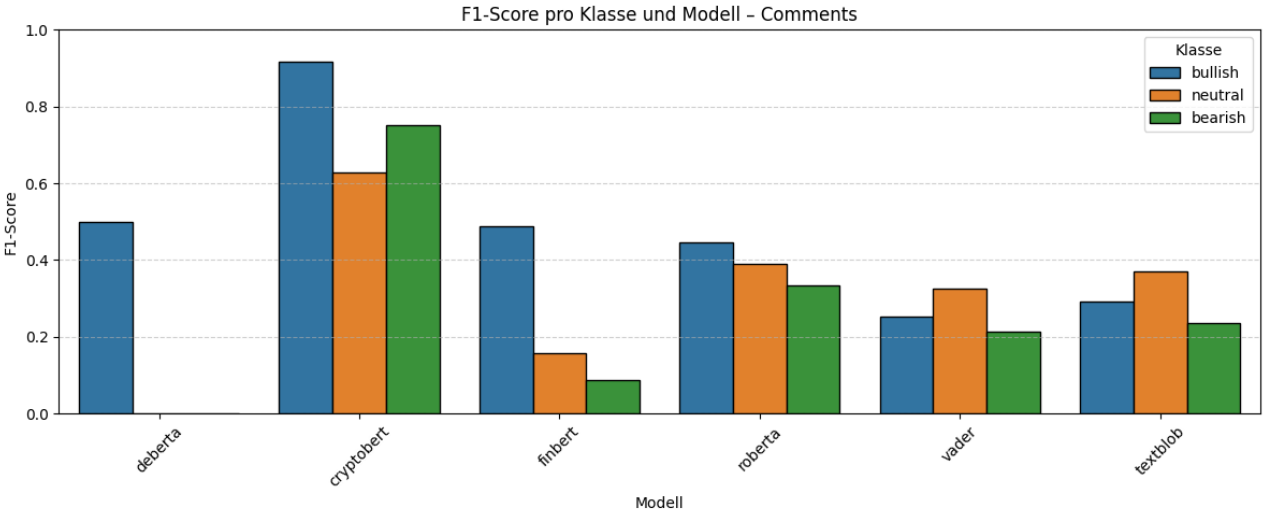

Abbildung 6: F1-Scores der Ausgangsmodelle auf Reddit-Kommentaren

Diese Ergebnisse verdeutlichen zwei zentrale Punkte:

1. Lexikonbasierte Modelle (VADER, TextBlob) erreichen bei kurzen Kommentaren nur niedrige F1-Scores, insbesondere bei bearish.

2. Transformer-Modelle zeigen ohne Anpassung eine hohe Streuung – nur CryptoBERT liefert bereits überdurchschnittliche Werte, vor allem bei bullish und bearish.

4.2 Fine-Tuning Ergebnisse

Nach gezieltem Fine-Tuning der jeweils besten Modelle konnten deutliche Leistungssteigerungen erzielt werden.

CryptoBERT wurde auf dem großen externen Kommentardatensatz feinjustiert. Die Ergebnisse nach zehn Epochen mit einer Lernrate von 1e-5 zeigen eine starke Gesamtleistung (siehe Tabelle 1):

- F1-Score (macro): 0.94
- Accuracy: 0.94
- Stabile Ergebnisse in allen Klassen:
  bullish (0.95), neutral (0.91), bearish (0.96)

Das Modell zeigt eine gleichmäßige Klassifikation über alle Sentiment-Kategorien hinweg. Besonders hervorzuheben ist die zuverlässige Erkennung von bearish-Kommentaren, die ohne Feintuning zuvor deutlich schlechter ausfiel.

FinBERT wurde auf einem selbst gelabelten Datensatz mit rund 1.280 Reddit-Posts trainiert. Auch hier erfolgte ein Vergleich verschiedener Lernraten über mehrere Epochen. Die besten Ergebnisse ergaben sich mit einer Lernrate von 2e-5:

- F1-Score (macro): 0.89
- Accuracy: 0.93
- Klassenergebnisse:
  bullish (0.85), neutral (0.95), bearish (0.89)

Im Vergleich zur Baseline ist das eine signifikante Verbesserung. Trotz kleinerer Trainingsmenge erreicht das Modell eine hohe Treffsicherheit – insbesondere bei der schwierigen Klasse bullish, die im Testset stark unterrepräsentiert war.

4.3 Lernratenvergleich

Abbildung 7 und 8 zeigen die Lernkurven des F1-Scores (macro) auf dem Validierungsset für verschiedene Lernraten:

Bei CryptoBERT (Abb. 7) lieferten die Lernraten 1e-5, 5e-6 und 3e-6 die besten Ergebnisse – mit stabilen und anhaltend hohen F1-Werten ab Epoche 4.

Die niedrigste Lernrate (1e-6) blieb konstant, aber unter dem Leistungsniveau der übrigen.

2e-5 zeigte eine auffällige Instabilität mit Leistungsabfall ab Epoche 3 und wurde verworfen.

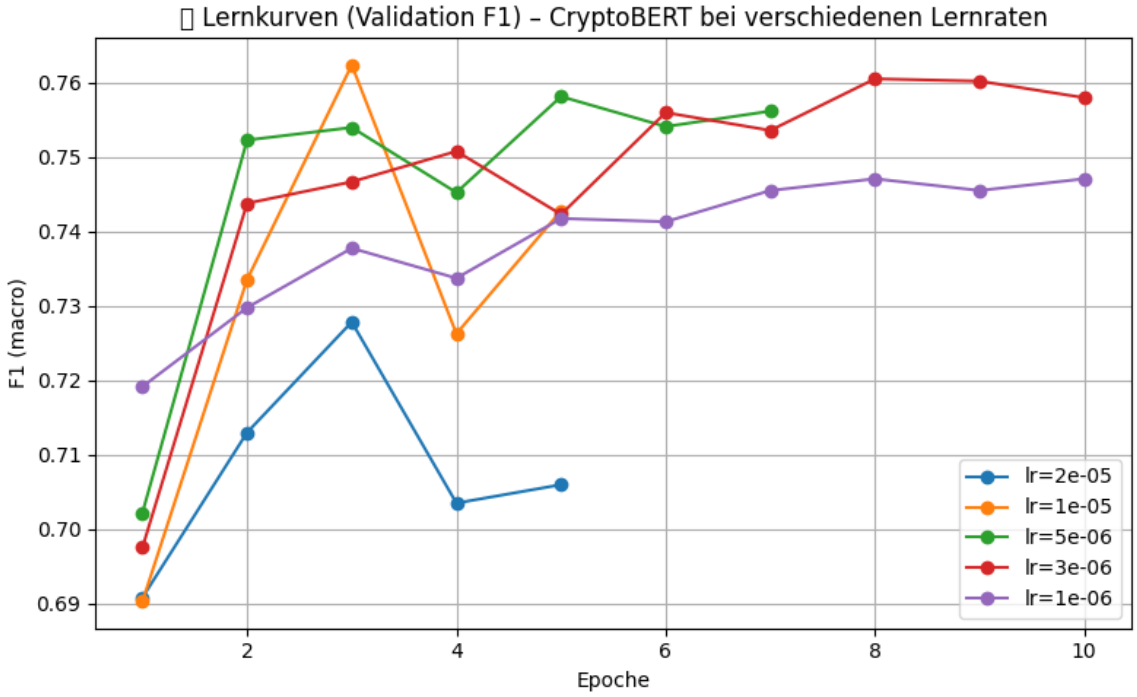

Abbildung 7: Lernkurven(F1 macro) für CryptoBERT bei verschiedenen Lernratenauf dem Kommentar-Datensatz

Bei FinBERT (Abb. 8) erzielte die Lernrate 2e-5 bereits ab Epoche 2 sehr hohe F1-Werte über 0.90 und blieb über den gesamten Trainingsverlauf hinweg stabil. Auch 1e-5 zeigte gute Resultate, wenn auch leicht schwankend.

Niedrigere Lernraten wie 3e-6, 5e-6 und 1e-6 blieben deutlich darunter und erreichten nur moderate Werte.

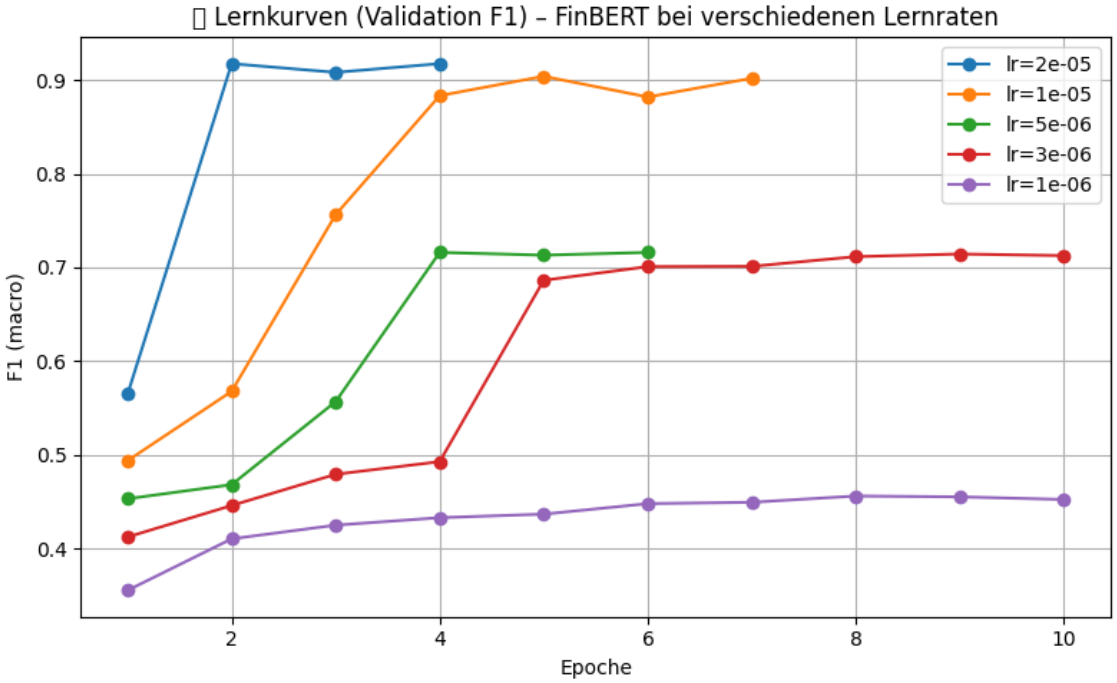

Abbildung 8: Lernkurven (F1 macro) für FinBERT bei verschiedenen Lernraten auf dem Post-Datensatz

Diese Beobachtungen bestätigen die Hyperparameter-Sensitivität von Transformer-Modellen, insbesondere bei kleineren oder unbalancierten Datensätzen – ein bekanntes Phänomen, das in der Literatur regelmäßig thematisiert wird (Huyen, 2022, Kapitel 7).

4.4 Gesamtbewertung

Die besten Ergebnisse wurden mit CryptoBERT nach Fine-Tuning erzielt. Die sehr hohe Klassifikationsgüte bei Kommentaren (F1 macro: 0.94) bestätigt, dass domänenspezifisches Vortraining und eine passende Datenstruktur zentrale Erfolgsfaktoren für Sentimentmodelle im Krypto Umfeld sind. Auch FinBERT erreichte nach Feintuning ein starkes Ergebnis (F1 macro: 0.89), blieb aber leicht hinter CryptoBERT zurück – unter anderem aufgrund der geringeren Datenmenge und der stärkeren Klassenungleichverteilung im Trainingssatz.
Die Ergebnisse legen nahe, dass in zukünftigen Projekten der Fokus verstärkt auf:

- skalierbare Datenbeschaffung (z. B. GPT-unterstütztes Labeling oder Bootstrapping).
- strukturierte Textaufteilung nach Format (Posts vs. Comments).
- datengetriebene Hyperparameteroptimierung gelegt werden sollte.

5 Fazit

Ziel dieser Arbeit war es, verschiedene NLP-Modelle zur Sentimentanalyse von Reddit-Beiträgen rund um das Thema Kryptowährungen systematisch zu untersuchen. Verglichen wurden klassische, lexikonbasierte Verfahren sowie moderne, transformerbasierte Sprachmodelle. In einem zweiten Schritt wurden die leistungsfähigsten Modelle durch gezieltes Fine-Tuning weiter optimiert. Untersucht wurde dabei insbesondere, wie zuverlässig sich Aussagen aus Reddit, oft ironisch, informell oder sehr kurz, den Kategorien bullish, neutral und bearish zuordnen lassen.
Die Ergebnisse zeigen deutlich: Transformerbasierte Modelle liefern bei entsprechender Anpassung klar bessere Resultate als klassische Methoden. Besonders CryptoBERT überzeugte durch eine hohe und ausgewogene Klassifikationsgüte auf dem Kommentar-Datensatz. Nach dem Fine-Tuning erreichte es einen macro-F1-Score von 0.94 bei einer Accuracy von 0.94 mit stabilen Ergebnissen in allen drei Sentimentklassen. Das Modell konnte insbesondere bearish-Kommentare präzise erkennen, ein Bereich, in dem klassische Modelle sowie unbehandelte Transformer zuvor schwächelten.
Auch FinBERT konnte nach dem Fine-Tuning deutlich zulegen und erreichte einen macro-F1-Score von 0.89. Trotz kleinerer Trainingsbasis und starker Klassenungleichheit schnitt es

bei längeren Reddit-Posts solide ab. Vor allem die Erkennung von bullish-Posts wurde durch das Training deutlich verbessert. Damit zeigt sich, dass auch mit begrenzten Ressourcen praxistaugliche Modelle aufgebaut werden können, sofern Datenstruktur, Modellwahl und Hyperparameter gut aufeinander abgestimmt sind.

5.1 Ausblick

Die Ergebnisse dieser Arbeit bilden eine fundierte Grundlage für den weiteren Einsatz transformerbasierter Sentimentmodelle im Finanzbereich. Eine zentrale Perspektive liegt in der skalierbaren Erweiterung der Trainingsdaten, etwa durch semi-supervised Learning, Active Learning oder automatisiertes Pseudo-Labeling. Solche Verfahren können helfen, unannotierte Reddit-Beiträge systematisch zu erschließen und die Datenbasis deutlich zu verbreitern (Huyen, 2022, Kapitel 10).

Ein weiterer Schritt wäre die Integration erklärbarer KI-Verfahren wie LIME oder SHAP, um die Entscheidungen der Modelle nachvollziehbarer zu machen, insbesondere bei mehrdeutigen, ironischen oder widersprüchlichen Aussagen. Dies könnte die Transparenz und das Vertrauen in die Modellvorhersagen stärken.

Langfristig wäre auch der Aufbau eines interaktiven Dashboards denkbar, das Sentimentdaten aus Reddit automatisiert auswertet und visualisiert, etwa für Investorinnen und Investoren, Analystinnen und Analysten oder Forschende. In Kombination mit Marktdaten ließe sich so ein praktisches Frühwarnsystem für Stimmungslagen im Krypto-Sektor entwickeln.

Zusammenfassend zeigt diese Arbeit: Transformerbasierte Sentimentmodelle können bei sorgfältiger Datenaufbereitung und gezieltem Feintuning einen wesentlichen Beitrag zur automatisierten Stimmungsanalyse im Social-Media-Bereich leisten. Trotz bestehender Herausforderungen bieten sie bereits heute leistungsfähige und differenzierte Ansätze zur Erfassung kollektiver Meinungen.

Literaturverzeichnis

Bramer, M. (2007). *Principles of data mining* (Vol. 180). London: Springer.

Downey, A. B., & Lang, J. W. (2024). *Python lernen mit KI-Tools: Einstieg in die Programmierung mit KI-Unterstützung*. o'Reilly Verlag GmbH & Co. KG.

Huyen, C. (2022). *Designing machine learning systems: An iterative process for production-ready applications*. O'Reilly Media.

IEEE (2023). *Sentiment analysis of crypto-related Reddit posts using transformer models*.