scrape

February 28, 2025

0.1 1 Einleitung

Diese Dokumentation beschreibt die Entwicklung einer datengetriebenen Pipeline zur Analyse der Stimmungslage (Sentiment) von Kryptowährungen basierend auf Reddit-Daten. Ziel ist es, durch automatisierte Datenerfassung, Verarbeitung und Analyse wertvolle Einblicke in Markttrends zu gewinnen. Die Implementierung umfasst die wöchentliche Erfassung von Reddit-Posts und -Kommentaren, deren Bereinigung und anschließende Sentiment-Analyse. Die Ergebnisse werden durch kontinuierliche Integrationsprozesse mittels Jenkins automatisiert verarbeitet und in einem interaktiven Dashboard mit Streamlit visualisiert.

0.1.1 Relevanz

Kryptowährungen unterliegen starken Kursschwankungen, die oft durch öffentliche Meinungen und Diskussionen in sozialen Netzwerken begleitet werden. Eine systematische Analyse dieser Stimmungen kann helfen: - Markttrends frühzeitig zu erkennen, - Investitionsentscheidungen zu unterstützen, - Risiken besser zu bewerten.

0.1.2 Zielsetzung

- Automatisierte Erfassung und Speicherung von relevanten Reddit-Diskussionen über Kryptowährungen.
- Bereinigung und Vorverarbeitung der gesammelten Daten, um eine hohe Datenqualität zu gewährleisten.
- Anwendung von Sentiment-Analyseverfahren zur Kategorisierung und Quantifizierung von Stimmungen.
- Automatisierte Bereitstellung der Ergebnisse in einem Dashboard zur kontinuierlichen Überwachung.

0.2 2 Einrichtung des Git-Repositories

0.2.1 Vorgehensweise:

1. **Erstellung eines öffentlichen Repositories**

```
git init
git remote add origin <URL>
git add .
git commit -m "Initial commit"
git push -u origin main
```

2. **Erzeugung eines Personal Access Tokens für Jenkins**

- Navigiere zu GitHub Personal Access Tokens
- Erstelle ein Token mit den erforderlichen Berechtigungen für **Repo** und **Workflows**
- Speichere das Token sicher und hinterlege es, du brauchst es später um Authentifizierungsprozesse für CI/CD zu ermöglichen.

0.3 3 Einrichtung der Reddit API

0.3.1 Vorgehensweise:

1. **Registrierung einer Anwendung auf Reddit**
   - Besuche die Reddit Developer Console
   - Erstelle eine neue Anwendung und wähle den Typ **script**
   - Trage eine beliebige **App-Name**, **Beschreibung** und eine **Redirect URL** (z. B. `http://localhost:8080`) ein
   - Notiere dir die generierte **Client ID** und **Client Secret**
2. **Umgebungsvariablen setzen**
   - Erstelle eine `.env` Datei und speichere die Anmeldeinformationen:

```
CLIENT_ID='deine_client_id'
CLIENT_SECRET='dein_client_secret'
USER_AGENT='dein_user_agent'
```

0.4 4 Einrichtung des Jupyter Notebooks

Um die Datenverarbeitung und Analyse interaktiv durchzuführen, wird ein **Jupyter Notebook** verwendet.

0.4.1 Installation der benötigten Python-Bibliotheken

Zunächst müssen alle relevanten Pakete installiert werden:

```
pip install praw pandas psaw python-dotenv torch transformers tqdm
```

0.4.2 Laden und Initialisieren der Reddit API

Das folgende Skript lädt die API-Zugangsdaten aus der `.env` Datei und stellt eine Verbindung zur Reddit API her:

[8]:
```
# Importiere die benötigten Bibliotheken
import praw
import pandas as pd
from datetime import datetime, timedelta, timezone
import os
import psaw as ps
from dotenv import load_dotenv
import time
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch.nn.functional as F
import torch
from tqdm import tqdm
```

```
from psaw import PushshiftAPI
from praw.exceptions import APIException
import requests
```

[9]:
```
# Lade die .env-Datei
dotenv_loaded = load_dotenv("zugang_reddit.env")  # Falls die Datei anders␣
 ↪heißt, anpassen
print(f".env geladen? {dotenv_loaded}")
```

```
.env geladen? True
```

[10]:
```
# Verbindung zur Reddit API
reddit = praw.Reddit(
    client_id=os.getenv("CLIENT_ID"),
    client_secret=os.getenv("CLIENT_SECRET"),
    user_agent=os.getenv("USER_AGENT")
)

print("Reddit API erfolgreich verbunden!")
```

```
Reddit API erfolgreich verbunden!
```

0.4.3 Testabfrage von Reddit-Posts

Um die Verbindung zu überprüfen, werden die fünf heißesten Posts aus dem Subreddit `CryptoCurrency` abgerufen:

[11]:
```
try:
    subreddit = reddit.subreddit("CryptoCurrency")
    for post in subreddit.hot(limit=5):
        print(f"Title: {post.title}, Score: {post.score}, URL: {post.url}")
except Exception as e:
    print(f"Fehler beim Abrufen der Posts: {e}")
```

```
Title: Daily Crypto Discussion - February 28, 2025 (GMT+0), Score: 23, URL: http
s://www.reddit.com/r/CryptoCurrency/comments/1izudyx/daily_crypto_discussion_feb
ruary_28_2025_gmt0/
Title: Thanks Crypto, Score: 4940, URL: https://i.redd.it/67mfttb6mple1.jpeg
Title: Genuinely, what happened to the "Crypto President"?, Score: 266, URL: htt
ps://www.reddit.com/r/CryptoCurrency/comments/1izz8iy/genuinely_what_happened_to
_the_crypto_president/
Title: US Act to Ban Political Crypto Tokens like TRUMP, Score: 1790, URL:
https://crypto-economy.com/us-act-to-ban-political-crypto-tokens-like-trump/
Title: Mining Bitcoin is gambling with extra steps, Score: 1774, URL:
https://i.redd.it/ea2h5o82fple1.png
```

Dieser Codeblock stellt sicher, dass die Verbindung zur Reddit API erfolgreich funktioniert. Falls es zu Fehlern kommt, sollten die API-Zugangsdaten überprüft werden.

0.4.4 Definition der Kryptowährungen und Subreddits

Um die relevante Diskussion zu Kryptowährungen zu analysieren, wird eine Liste von Kryptowährungen mit ihren Symbolen sowie eine Auswahl an Subreddits definiert.

Die folgende Liste enthält die wichtigsten Kryptowährungen, die in der Analyse berücksichtigt werden:

[12]:
```
crypto_terms = {
    #  Top Coins
    "Bitcoin": ["bitcoin", "btc"],
    "Ethereum": ["ethereum", "eth"],
    "Wrapped Ethereum": ["wrapped ethereum", "weth"],
    "Solana": ["solana", "sol"],
    "Avalanche": ["avalanche", "avax"],
    "Polkadot": ["polkadot", "dot"],
    "Near Protocol": ["near protocol", "near"],
    "Polygon": ["polygon", "matic"],
    "XRP": ["xrp", "ripple"],
    "Cardano": ["cardano", "ada"],
    "Cronos": ["cronos", "cro"],
    "Vulcan Forged PYR": ["vulcan forged", "pyr"],
    "Chiliz": ["chiliz", "chz"],
    "Illuvium": ["illuvium", "ilv"],
    "Ronin": ["ronin", "ron"],
    "Band Protocol": ["band protocol", "band"],
    "Optimism": ["optimism", "op"],
    "Celestia": ["celestia", "tia"],
    "Numerai": ["numerai", "nmr"],
    "Aethir": ["aethir", "ath"],
    "Sui": ["sui"],
    "Hyperliquid": ["hyperliquid", "hyp"],
    "Robinhood Coin": ["robinhood", "hood"],
    "Trump Coin": ["trump coin"],
    "USD Coin": ["usd coin", "usdc"],
    "Binance Coin": ["binance", "bnb"],
    "Litecoin": ["litecoin", "ltc"],
    "Dogecoin": ["dogecoin", "doge"],
    "Tron": ["tron", "trx"],
    "Aave": ["aave"],
    "Hedera": ["hedera", "hbar"],
    "Filecoin": ["filecoin", "fil"],
    "Cosmos": ["cosmos", "atom"],
    "Gala": ["gala"],
    "The Sandbox": ["sandbox", "sand"],
    "Audius": ["audius", "audio"],
    "Render": ["render", "rndr"],
    "Kusama": ["kusama", "ksm"],
```

```
    "VeChain": ["vechain", "vet"],
    "Chainlink": ["chainlink", "link"],
    "Berachain": ["berachain", "bera"],
    "TestCoin": ["testcoin", "test"],

    #  Meme-Coins
    "Shiba Inu": ["shiba inu", "shib"],
    "Pepe": ["pepe"],
    "Floki Inu": ["floki inu", "floki"],
    "Bonk": ["bonk"],
    "Wojak": ["wojak"],
    "Mog Coin": ["mog"],
    "Doge Killer (Leash)": ["leash"],
    "Baby Doge Coin": ["baby doge", "babydoge"],
    "Degen": ["degen"],
    "Toshi": ["toshi"],
    "Fartcoin": ["fartcoin"],
    "Banana": ["banana"],
    "Kabosu": ["kabosu"],
    "Husky": ["husky"],
    "Samoyedcoin": ["samoyedcoin", "samo"],
    "Milkbag": ["milkbag"],

    #  New Coins
    "Arbitrum": ["arbitrum", "arb"],
    "Starknet": ["starknet", "strk"],
    "Injective Protocol": ["injective", "inj"],
    "Sei Network": ["sei"],
    "Aptos": ["aptos", "apt"],
    "EigenLayer": ["eigenlayer", "eigen"],
    "Mantle": ["mantle", "mnt"],
    "Immutable X": ["immutable x", "imx"],
    "Ondo Finance": ["ondo"],
    "Worldcoin": ["worldcoin", "wld"],
    "Aerodrome": ["aerodrome", "aero"],
    "Jupiter": ["jupiter", "jup"],
    "THORChain": ["thorchain", "rune"],
    "Pendle": ["pendle"],
    "Kujira": ["kujira", "kuji"],
    "Noble": ["noble"],
    "Stride": ["stride", "strd"],
    "Dymension": ["dymension", "dym"],
    "Seamless Protocol": ["seamless", "seam"],
    "Blast": ["blast"],
    "Merlin": ["merlin"],
    "Tapioca": ["tapioca"],
    "Arcadia Finance": ["arcadia"],
```

```
    "Notcoin": ["notcoin", "not"],
    "Omni Network": ["omni"],
    "LayerZero": ["layerzero", "lz"],
    "ZetaChain": ["zetachain", "zeta"],
    "Friend.tech": ["friendtech"]
}
```

Zur Analyse der Diskussionen werden Subreddits verwendet, die sich intensiv mit Kryptowährungen und deren Marktbewegungen beschäftigen:

[13]:
```
subreddits = [
    "CryptoCurrency",  # Allgemeine Diskussionen über Kryptowährungen
    "CryptoMarkets",   # Diskussionen über den Kryptomarkt und Preisbewegungen
    "CryptoTrading",   # Fokus auf Trading-Strategien und Analysen
    "Altcoin",         # Diskussionen über Altcoins (alle Kryptowährungen außer␣
 ↪Bitcoin)
    "DeFi",            # Decentralized Finance (DeFi) und Projekte
    "BitcoinBeginners",# Für Anfänger in der Krypto-Welt
    "cryptotechnology", # Fokus auf die zugrunde liegende Blockchain-Technologie
    "cryptocurrencies", # Allgemeine Diskussionen über Kryptowährungen
    "Satoshistreetsbets", # Krypto-Wetten und Spekulationen
    "Binance",        # Diskussionen über die Binance-Plattform
    "Bitcoin",          # Diskussionen über Bitcoin
    "ethtrader"     # Generelle Diskussionen über Crypto
]
```

Diese Listen werden später verwendet, um relevante Beiträge und Kommentare aus diesen Subreddits zu extrahieren.

0.4.5 Scraping von Posts und Kommentaren

Die folgende Funktion ermöglicht es, gezielt Reddit-Posts und deren Kommentare für bestimmte Kryptowährungen zu extrahieren. Dabei werden sowohl der vollständige Name als auch das Kürzel (case-insesitive) der Kryptowährung als Suchbegriffe genutzt.

[14]:
```
  #  Funktion zum Scrapen von Posts und Kommentaren mit Backoff
def scrape_reddit(start_date, end_date, mode="initial"):
    start_timestamp = int(start_date.timestamp())
    end_timestamp = int(end_date.timestamp())

    posts = []
    comments = []
    post_ids = set()
    request_count = 0  # Zählt die Anzahl der Requests

    for crypto_name, search_terms in crypto_terms.items():
        for subreddit_name in subreddits:
            subreddit = reddit.subreddit(subreddit_name)
```

```
            print(f"  Suche nach {crypto_name} in r/{subreddit_name}...")

            #  Suche nach allen relevanten Begriffen mit `.search()`
            for search_term in search_terms:
                for post in subreddit.search(query=search_term, sort="new",␣
↪limit=None):
                    if start_timestamp <= post.created_utc <= end_timestamp and␣
↪post.id not in post_ids:
                        post_ids.add(post.id)
                        created_dt = datetime.utcfromtimestamp(post.created_utc)

                        posts.append({
                            'post_id': post.id,
                            'crypto': crypto_name,
                            'search_term': search_term,
                            'subreddit': subreddit_name,
                            'title': post.title,
                            'author': str(post.author),
                            'date': created_dt.date().isoformat(),
                            'time': created_dt.time().isoformat(),
                            'score': post.score,
                            'num_comments': post.num_comments,
                            'selftext': post.selftext
                        })
                        print(f"  Post gefunden: {post.title} (Suchbegriff:␣
↪{search_term})")

                        #  Kommentare sammeln (mit Rate-Limit-Schutz)
                        try:
                            post.comments.replace_more(limit=0)
                            for comment in post.comments.list():
                                created_dt = datetime.utcfromtimestamp(comment.
↪created_utc)

                                comments.append({
                                    'post_id': post.id,
                                    'comment_id': comment.id,
                                    'author': str(comment.author),
                                    'date': created_dt.date().isoformat(),
                                    'time': created_dt.time().isoformat(),
                                    'score': comment.score,
                                    'selftext': comment.body
                                })
                        except praw.exceptions.APIException as e:
                            if "RATELIMIT" in str(e):
                                print(f"  Reddit API-Limit erreicht. Warte 60␣
↪Sekunden...")
                                time.sleep(60)  # Wartezeit erhöhen
```

```
                            else:
                                print(f"  Fehler beim Abrufen der Kommentare:␣
↪{e}")

                    #  Nach jeder `post.comments.list()` Anfrage prüfen, ob␣
↪eine Pause nötig ist
                    request_count += 1
                    if request_count % 50 == 0:  # Nach 50 Requests eine kurze␣
↪Pause
                        wait_time = 10  # Standard-Wartezeit
                        print(f"  Warte {wait_time} Sekunden, um Rate-Limit zu␣
↪vermeiden...")
                        time.sleep(wait_time)

    # In DataFrames umwandeln
    df_posts = pd.DataFrame(posts)
    df_comments = pd.DataFrame(comments)

    print(f"  Scrape abgeschlossen: {len(df_posts)} Posts & {len(df_comments)}␣
↪Kommentare gefunden.")
    return df_posts, df_comments
```

0.4.6 Durchführung des Scrapes

Ein Beispielaufruf für die Funktion über die letzten drei Monate:

[15]:
```
# #  Starte den Scraper für die letzten 3 Monate
# start_of_period = datetime(2024, 11, 1)  # Startzeitpunkt
# now = datetime.now()  # Aktueller Zeitpunkt
# print("  Starte den Scraper für die letzten 3 Monate...")
# df_posts, df_comments = scrape_reddit(start_of_period, now)

# print("  Daten erfolgreich gespeichert & bereit für weitere Analysen.")
```

Bei dem wöchentlichen Scrape sieht der Aufruf wie folgt aus:

[ ]:
```
# Aktuelle Zeit als datetime-Objekt
now = datetime.now()
last_week = now - timedelta(days=7)  # 7 Tage zurück

print("  Starte den wöchentlichen Scrape...")

# Aufruf der Scraper-Funktion mit datetime-Objekten
df_posts, df_comments = scrape_reddit(last_week, now, mode="weekly")

# Beispiel: Lokale Weiterverarbeitung
print("Daten können jetzt bereinigt werden...")
```

```
 Starte den wöchentlichen Scrape…
 Suche nach Bitcoin in r/CryptoCurrency…
 Post gefunden: Are gold based stablecoins the future? (Suchbegriff: bitcoin)
 Post gefunden: What's the future of memecoins/crypto outside of btc/eth?
(Suchbegriff: bitcoin)
 Post gefunden: Bitcoin sees largest weekly drop in market cap since inception
(Suchbegriff: bitcoin)
 Post gefunden: Introducing Tap-to-Pay for Crypto, Expanding Payment
Accessibility - Breakdown (Suchbegriff: bitcoin)
 Post gefunden: Bitcoin ETFs Have Shed More Than $2.4 Billion So Far This Week
(Suchbegriff: bitcoin)
 Post gefunden: Texas Bitcoin Reserve Bill Passes Committee With 9-0 Vote,
What’s Next? (Suchbegriff: bitcoin)
 Post gefunden: Scam or real? Trianz.life (Suchbegriff: bitcoin)
 Post gefunden: Bitcoin whiplash (Suchbegriff: bitcoin)
 Post gefunden: Recent buyers Bitcoin (BTC) have realized $2.16B in losses
(Suchbegriff: bitcoin)
 Post gefunden: 'Extreme fear': Crypto Fear & Greed Index hits multi-year low,
as bitcoin sinks below $86,000 (Suchbegriff: bitcoin)
 Post gefunden: Mining Bitcoin is gambling with extra steps (Suchbegriff:
bitcoin)
 Post gefunden: Litecoin and Dogecoin are examples why MicroStrategy could
easily go bust. (Suchbegriff: bitcoin)
 Post gefunden: Astrology 'Pro' predicts Bitcoin Will Crash & Rebound - Might
as well trust this as much as other analysts.. (Suchbegriff: bitcoin)
 Post gefunden: Metaplanet to Raise ¥2.0 Billion to Buy More Bitcoin (BTC) -
They Really Want To Buy The Dip (Suchbegriff: bitcoin)
 Post gefunden: He didn't buy the dip  (Suchbegriff: bitcoin)
 Post gefunden: Bitcoin Fear And Greed Index At 10, Which Is Lower Than FTX
Collapse (Suchbegriff: bitcoin)
```

Ich habe zunächst den einmaligen Scrape durchlaufen lassen und anschliessend den wöchentlichen automatisiert, dazu kommen wir später.

0.4.7 Datenbereinigung und Vorverarbeitung

Diese Funktion: - Entfernt Duplikate in Posts und Kommentaren. - Handhabt fehlende Werte, um Datenverluste zu vermeiden. - Konvertiert und strukturiert Zeitstempel für bessere Nachvollziehbarkeit. - Zu kurze Kommentare werden entfernt. - Entfernt Kommentare von Accounts mit übermäßig vielen Kommentaren.

[ ]:
```python
def clean_data(df_posts, df_comments, comment_threshold=500, min_length=5):
    # 1. Duplikate entfernen
    df_posts = df_posts.drop_duplicates(subset=["post_id"])
    df_comments = df_comments.drop_duplicates(subset=["comment_id"])

    # 2. Fehlende Werte behandeln
```

```
    df_posts['selftext'] = df_posts['selftext'].fillna('')  # Fehlende␣
↪Posttexte auffüllen
    df_comments['selftext'] = df_comments['selftext'].fillna('')  # Fehlende␣
↪Kommentare auffüllen

    # 3. Entferne Nutzer (Bots) mit übermäßigen Kommentaren
    comment_counts = df_comments["author"].value_counts()
    frequent_users = comment_counts[comment_counts > comment_threshold].index ␣
↪# Nutzer über Grenze
    df_comments = df_comments[~df_comments["author"].isin(frequent_users)]

    # 4. Entferne zu kurze Kommentare
    df_comments = df_comments[df_comments['selftext'].str.len() >= min_length]

    print(f" Daten bereinigt: {df_comments.shape[0]} Kommentare übrig (nach␣
↪Spam-Filter & Länge > {min_length}).")

    return df_posts, df_comments
```

0.4.8 Anwendung der Bereinigungsfunktion

Nach der Extraktion der Daten wird die Bereinigungsfunktion auf die Posts und Kommentare angewendet:

[ ]:
```
# Bereinigen der Daten
df_posts_clean, df_comments_clean = clean_data(df_posts, df_comments,␣
↪comment_threshold=500, min_length=5)


# Überprüfen, wie viele Einträge übrig sind
print(f"Bereinigte Posts: {len(df_posts_clean)}")
print(f"Bereinigte Kommentare: {len(df_comments_clean)}")
```

```
 Daten bereinigt: 26232 Kommentare übrig (nach Spam-Filter & Länge > 5).
Bereinigte Posts: 763
Bereinigte Kommentare: 26232
```

Diese Schritte stellen sicher, dass nur relevante, qualitativ hochwertige Daten in der Pipeline weiterverarbeitet werden.

0.4.9 Sentiment-Analyse der Kommentare

Nach der Bereinigung der Daten wird das Sentiment für die gesammelten Kommentare mithilfe eines vortrainierten Modells analysiert.

Verwendetes Sentiment-Analyse-Modell Das Modell "ElKulako/crypto-bert" stammt von ElKulako und wurde speziell für die Analyse von Kryptowährungs-Diskussionen entwickelt. Es basiert auf der BERT-Architektur und klassifiziert Texte als bullish (positiv), neutral oder bearish

(negativ). Das Modell wurde auf umfangreichen Finanz- und Krypto-spezifischen Daten trainiert, wodurch es sich besonders gut für die Analyse von Reddit-Kommentaren eignet, die sich mit dem Krypto-Markt befassen.

[ ]:
```
# GPU nutzen, falls verfügbar sonst weglassen
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"  Verwende Gerät: {device}")

#   CryptoBERT-Modell laden
MODEL_NAME = "ElKulako/cryptobert"
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
model = AutoModelForSequenceClassification.from_pretrained(MODEL_NAME).
 ↪to(device)
model.eval()  # Setzt das Modell in den Evaluationsmodus


#   Funktion zur Sentiment-Analyse (Optimiert für Batch-Prozesse)
def analyze_sentiment_batch(texts, batch_size=32):
    """Effiziente GPU-gestützte Sentiment-Analyse mit CryptoBERT für eine Liste␣
 ↪von Texten."""
    results = []

    # Ersetze leere Einträge durch "neutral"
    texts = [t if isinstance(t, str) and t.strip() != "" else "neutral" for t␣
 ↪in texts]

    # Batchweise Verarbeitung
    for i in tqdm(range(0, len(texts), batch_size), desc="  Analysiere␣
 ↪Sentiments"):
        batch_texts = texts[i : i + batch_size]

        # Tokenisierung (mit Padding für Performance)
        inputs = tokenizer(batch_texts, return_tensors="pt", truncation=True,␣
 ↪max_length=512, padding=True).to(device)

        # Vorhersage mit CryptoBERT
        with torch.no_grad():
            outputs = model(**inputs)

        scores = F.softmax(outputs.logits, dim=1)
        labels = ["bearish", "neutral", "bullish"]

        # Ergebnisse speichern
        for i in range(len(batch_texts)):
            sentiment = labels[torch.argmax(scores[i]).item()]
            confidence = scores[i].max().item()
            results.append((sentiment, confidence))
```

```
    return results

#  Sentiment für **Posts** berechnen
tqdm.pandas()  # Fortschrittsanzeige aktivieren
df_posts_clean["full_text"] = df_posts_clean["title"] + " " +␣
 ↪df_posts_clean["selftext"].fillna("")
df_posts_clean[["sentiment", "sentiment_confidence"]] = pd.DataFrame(
    analyze_sentiment_batch(df_posts["full_text"].tolist()), index=df_posts.
 ↪index
)

#  Sentiment für **Kommentare** berechnen
df_comments_clean["full_text"] = df_comments_clean["selftext"].fillna("")
df_comments_clean[["sentiment", "sentiment_confidence"]] = pd.DataFrame(
    analyze_sentiment_batch(df_comments_clean["full_text"].tolist()),␣
 ↪index=df_comments_clean.index
)

#  Ergebnisse anzeigen
print(f" Sentiment-Analyse abgeschlossen: {len(df_posts_clean)} Posts &␣
 ↪{len(df_comments_clean)} Kommentare bewertet.")
```

```
Verwende Gerät: cuda

Analysiere Sentiments: 100%|      | 24/24 [00:11<00:00,  2.07it/s]
Analysiere Sentiments: 100%|      | 820/820 [02:56<00:00,  4.64it/s]

Sentiment-Analyse abgeschlossen: 763 Posts & 26232 Kommentare bewertet.
```

0.4.10 Merging der Daten

Nachdem die Sentiment-Analyse durchgeführt wurde, werden die bereinigten Posts und Kommentare zusammengeführt.

[ ]:
```
posts = df_posts_clean.copy()
comments = df_comments_clean.copy()
```

[ ]:
```
posts.head()
```

[ ]:
```
   post_id   crypto search_term       subreddit  \
0  1izdutr  Bitcoin     bitcoin  CryptoCurrency
1  1iza5sg  Bitcoin     bitcoin  CryptoCurrency
2  1iz9hgb  Bitcoin     bitcoin  CryptoCurrency
3  1iz9fz2  Bitcoin     bitcoin  CryptoCurrency
4  1iz8vb1  Bitcoin     bitcoin  CryptoCurrency
```

```
                                                title                author  \
0                              He didn't buy the dip    rizzobitcoinhistory
1  Nearly 4% of the World Owns Bitcoin - Just the…             kirtash93
2  Bitcoin Fear And Greed Index At 10, Which Is L…       Funnyurolith61
3  BlackRock Bitcoin Fund Sees $420M Outflow as E…               Flygate
4                                  In the lavatory            thecryptos

         date      time  score  num_comments selftext  \
0  2025-02-27  11:44:17    181            26
1  2025-02-27  07:16:01      0             6
2  2025-02-27  06:28:00    212            70
3  2025-02-27  06:25:07      5             2
4  2025-02-27  05:48:10    279            23

                                           full_text sentiment  \
0                              He didn't buy the dip      bullish
1  Nearly 4% of the World Owns Bitcoin - Just the…   neutral
2  Bitcoin Fear And Greed Index At 10, Which Is L…   neutral
3  BlackRock Bitcoin Fund Sees $420M Outflow as E…   neutral
4                                  In the lavatory    neutral

   sentiment_confidence
0              0.635427
1              0.581964
2              0.651733
3              0.699549
4              0.385122
```

[ ]:
```
#  **Relevante Spalten für den Merge**
posts = posts[["post_id", "crypto", "search_term", "subreddit", "author",␣
 ↪"date", "time", "score", "full_text", "sentiment", "sentiment_confidence"]]
comments = comments[["post_id", "comment_id", "author", "date", "time",␣
 ↪"score", "full_text", "sentiment", "sentiment_confidence"]]

#  **Kommentare erben `crypto`, `search_term` und `subreddit` vom Post**
comments = comments.merge(df_posts[["post_id", "crypto", "search_term",␣
 ↪"subreddit"]], on="post_id", how="left")

#  `type`-Spalte für Unterscheidung hinzufügen
posts["comment_id"] = None  # Posts haben keine comment_id
posts["type"] = "post"
comments["type"] = "comment"

#  **Gemeinsame Spalten für den Merge**
common_columns = [
    "post_id", "comment_id", "type", "date", "time", "crypto",
```

```
    "search_term", "subreddit", "author","full_text","score", "sentiment",␣
  ↪"sentiment_confidence",
]

#  **Merging der Daten (Posts + Kommentare)**
df_merged = pd.concat([posts[common_columns], comments[common_columns]],␣
  ↪ignore_index=True)

#  Debugging: Überprüfung der Größe
print(f"  Merged Dataset: {df_merged.shape[0]} Einträge (Posts + Kommentare)")

#  Überprüfen, ob alles korrekt normalisiert wurde
print(df_merged.head())
```

```
 Merged Dataset: 26995 Einträge (Posts + Kommentare)
  post_id comment_id  type        date      time   crypto search_term  \
0  1izdutr       None  post  2025-02-27  11:44:17  Bitcoin     bitcoin
1  1iza5sg       None  post  2025-02-27  07:16:01  Bitcoin     bitcoin
2  1iz9hgb       None  post  2025-02-27  06:28:00  Bitcoin     bitcoin
3  1iz9fz2       None  post  2025-02-27  06:25:07  Bitcoin     bitcoin
4  1iz8vb1       None  post  2025-02-27  05:48:10  Bitcoin     bitcoin

       subreddit               author  \
0  CryptoCurrency  rizzobitcoinhistory
1  CryptoCurrency            kirtash93
2  CryptoCurrency       Funnyurolith61
3  CryptoCurrency              Flygate
4  CryptoCurrency           thecryptos

                                        full_text  score sentiment  \
0                          He didn't buy the dip      181   bullish
1  Nearly 4% of the World Owns Bitcoin - Just the…      0   neutral
2  Bitcoin Fear And Greed Index At 10, Which Is L…    212   neutral
3  BlackRock Bitcoin Fund Sees $420M Outflow as E…      5   neutral
4                                In the lavatory      279   neutral

   sentiment_confidence
0              0.635427
1              0.581964
2              0.651733
3              0.699549
4              0.385122
```

Dieser Schritt stellt sicher, dass alle relevanten Kommentare mit ihren zugehörigen Posts verknüpft sind sodass wir neben den normalisierten Post- und Kommentar-Tabelle eine fertige Tabelle für Analysen haben.

0.4.11 Export und Speicherung der Daten

Zweck der Speicherung Die exportierten Daten enthalten die gesammelten Reddit-Posts und Kommentare sowie deren Sentiment-Analyse. Durch die Speicherung in **Google Drive** wird eine einfache Automatisierung ermöglicht, sodass die Daten regelmäßig aktualisiert und für nachfolgende Analysen oder Machine-Learning-Prozesse verfügbar sind.

Implementierung des Exports Den Pfad definieren

[ ]:
```
# Setze den Pfad zu deinem Google Drive Ordner
DRIVE_PATH = "G:/Meine Ablage/reddit/"
POSTS_CSV = os.path.join(DRIVE_PATH, "reddit_posts.csv")
COMMENTS_CSV = os.path.join(DRIVE_PATH, "reddit_comments.csv")
MERGED_CSV = os.path.join(DRIVE_PATH, "reddit_merged.csv")
ORIGINAL_POSTS_CSV = os.path.join(DRIVE_PATH, "reddit_posts_original.csv")
ORIGINAL_COMMENTS_CSV = os.path.join(DRIVE_PATH, "reddit_comments_original.csv")
```

Die folgende Funktion speichert die extrahierten und verarbeiteten Daten als CSV-Dateien.

[ ]:
```
def append_to_csv(df_new, filename, key_column):
    """Hängt neue Daten an eine bestehende CSV an & entfernt Duplikate."""
    file_path = os.path.join(DRIVE_PATH, filename)

    try:
        # Falls Datei existiert, alte Daten einlesen
        if os.path.exists(file_path):
            df_existing = pd.read_csv(file_path, sep="|", encoding="utf-8-sig",␣
↪on_bad_lines="skip")

            #  Daten zusammenführen & Duplikate nach `key_column` entfernen␣
↪(neuere Werte behalten)
            df_combined = pd.concat([df_existing, df_new], ignore_index=True).
↪drop_duplicates(subset=[key_column], keep="last")
        else:
            df_combined = df_new  # Falls keine Datei existiert, neue Daten␣
↪direkt nutzen

        #  CSV speichern
        df_combined.to_csv(
            file_path,
            index=False,
            sep="|",
            encoding="utf-8-sig",
            lineterminator="\n"
        )
        print(f" Datei erfolgreich aktualisiert: {file_path}")

    except Exception as e:
```

```
        print(f"Fehler beim Speichern der Datei {filename}: {e}")

def export_to_drive(df_posts_clean, df_comments_clean, df_merged,df_posts,␣
 ↪df_comments):
    """Speichert Posts, Kommentare & die gemergte Datei mit Duplikat-Prüfung."""
    try:
        append_to_csv(df_posts_clean, "reddit_posts.csv", key_column="post_id")
        append_to_csv(df_comments_clean, "reddit_comments.csv",␣
 ↪key_column="comment_id")
        append_to_csv(df_merged, "reddit_merged.csv", key_column="comment_id") ␣
 ↪# Falls Kommentare entscheidend sind
        append_to_csv(df_posts, "reddit_posts_original.csv",␣
 ↪key_column="post_id")
        append_to_csv(df_comments, "reddit_comments_original.csv",␣
 ↪key_column="comment_id")
    except Exception as e:
        print(f"Fehler beim Export: {e}")
```

[ ]:
```
# Export-Funktion aufrufen
export_to_drive(df_posts_clean, df_comments_clean, df_merged,df_posts,␣
 ↪df_comments)
```

Vorteile der Speicherung in Google Drive

- **Automatisierung**: Die exportierten Daten können regelmäßig durch Skripte oder Jenkins-Jobs aktualisiert werden.
- **Zugänglichkeit**: Die gespeicherten Dateien können von verschiedenen Systemen oder Nutzern für Analysen oder Machine-Learning-Modelle verwendet werden.
- **Versionskontrolle**: Historische Daten können gespeichert werden, um Entwicklungen über die Zeit hinweg zu analysieren.

Dieser Schritt stellt sicher, dass die verarbeiteten Daten langfristig verfügbar bleiben und kontinuierlich für weitere Analysen genutzt werden können.

0.5 5 Automatisierung mit Jenkins

0.5.1 Einrichtung von Jenkins auf Ubuntu

Um den Scraping- und Analyseprozess zu automatisieren, wird Jenkins auf einem Ubuntu-Server eingerichtet.

1. **Installation von Jenkins auf Ubuntu**

```
sudo apt update
sudo apt install openjdk-11-jre
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenk
sudo apt update
sudo apt install jenkins
```

```
sudo systemctl start jenkins
sudo systemctl enable jenkins
```

Nach der Installation ist Jenkins unter `http://localhost:8080` erreichbar.

2. **Erstellen eines neuen Jobs in Jenkins**
   - Öffne die Jenkins Web-Oberfläche.
   - Erstelle einen neuen **Freestyle-Projekt**-Job.
   - Füge den **GitHub Personal Access Token** ein, um Zugriff auf das Repository zu erhalten.
3. **Konfiguration des Build-Schritts** Der folgende Shell-Befehl wird in den Build-Schritten des Jenkins-Jobs hinzugefügt, um das Notebook auszuführen:

```
# Aktiviere das Python-Environment
. /var/lib/jenkins/venv/bin/activate

# Lade die .env-Datei und setze die Variablen
set -a
. /var/lib/jenkins/workspace/reddit_crypto_scraper/.env
set +a

# Debugging: Prüfe, ob CLIENT_ID gesetzt wurde
echo "CLIENT_ID = $CLIENT_ID"

# Installiere Abhängigkeiten aus requirements.txt
pip install -r /var/lib/jenkins/workspace/reddit_crypto_scraper/requirements.txt

# Führe das Notebook mit papermill aus
papermill /var/lib/jenkins/workspace/reddit_crypto_scraper/notebooks/reddit-skript.ipynb /var/l
```

0.5.2 Vorteile der Jenkins-Automatisierung

- **Regelmäßige Ausführung**: Jenkins kann so konfiguriert werden, dass das Skript automatisch täglich oder wöchentlich ausgeführt wird.
- **Monitoring & Logging**: Jenkins speichert Logs jeder Ausführung und ermöglicht eine Fehleranalyse.
- **Reproduzierbarkeit**: Durch das Laden der `.env`-Datei und die Installation aller Abhängigkeiten ist jede Ausführung identisch.

Mit dieser Konfiguration ist die gesamte Sentiment-Analyse vollständig automatisiert und kann kontinuierlich aktualisiert werden.

0.6 6 Integrierung von Preisdaten

Dieses Skript ruft historische Preisdaten für verschiedene Kryptowährungen über die CoinGecko API ab. Die Kryptowährungen werden dabei direkt aus der bestehenden crypto_terms-Liste des Reddit-Scrapers übernommen, um Konsistenz zu gewährleisten. Die API-Abfragen erfolgen für die letzten 90 Tage, wobei ein Rate-Limit-Handling integriert wurde, um API-Sperren zu vermeiden. Die extrahierten Daten werden formatiert, in ein lesbares Datumsformat konvertiert und als CSV-

Datei gespeichert. Dies ermöglicht eine spätere Analyse der Preisentwicklungen im Zusammenhang mit den Reddit-Sentiment-Daten.

[ ]:
```
# CoinGecko IDs aus der zentralen Liste generieren
CRYPTO_LIST = {name: ids[0].replace(" ", "-") for name, ids in crypto_terms.
 ↪items()}

# Zeitraum für die letzten 90 Tage
DAYS = 90

# Speicherpfad (Google Drive Sync-Ordner)
DRIVE_PATH = "G:/Meine Ablage/reddit/"
OUTPUT_FILE = os.path.join(DRIVE_PATH, "crypto_prices.csv")

# Falls die Datei existiert, laden wir die bestehenden Daten
if os.path.exists(OUTPUT_FILE):
    existing_df = pd.read_csv(OUTPUT_FILE, sep="|", encoding="utf-8-sig",␣
 ↪parse_dates=["date"])
else:
    existing_df = pd.DataFrame(columns=["date", "crypto", "price"])

# Liste zur Speicherung der neuen Kursdaten
all_prices = []

# API-Abfrage für jede Kryptowährung
for crypto_name, crypto_id in CRYPTO_LIST.items():
    print(f" Abrufen von Kursdaten für {crypto_name}...")
    url = f"https://api.coingecko.com/api/v3/coins/{crypto_id}/market_chart?
 ↪vs_currency=usd&days={DAYS}"
    response = requests.get(url)

    if response.status_code == 200:
        data = response.json()
        prices = data["prices"]  # Liste von [timestamp, price]

        for price_data in prices:
            all_prices.append({
                "date": pd.to_datetime(price_data[0], unit="ms").date(),  #␣
 ↪Datum als YYYY-MM-DD
                "crypto": crypto_name,
                "price": price_data[1]
            })

        print(f" Erfolgreich abgerufen: {crypto_name}")
    elif response.status_code == 429:
        print(f" API-Limit erreicht für {crypto_name}. Warte 60 Sekunden...")
        time.sleep(60)
```

```
        continue
    else:
        print(f" Fehler beim Abrufen der API für {crypto_name}: {response.
 ↪status_code}")

    time.sleep(10)

# DataFrame mit neuen Daten erstellen
df_prices = pd.DataFrame(all_prices)

# Falls es bereits Daten gibt, Duplikate entfernen
if not existing_df.empty:
    combined_df = pd.concat([existing_df, df_prices])
    combined_df = combined_df.drop_duplicates(subset=["date", "crypto"],␣
 ↪keep="last")
else:
    combined_df = df_prices

# Daten speichern (ersetzen, falls Datei schon existiert)
combined_df.to_csv(OUTPUT_FILE, sep="|", encoding="utf-8-sig", index=False)

print(f" Kursdaten gespeichert, Duplikate entfernt: {OUTPUT_FILE}")
```

```
Abrufen von Kursdaten für Bitcoin…
Erfolgreich abgerufen: Bitcoin
Abrufen von Kursdaten für Ethereum…
Erfolgreich abgerufen: Ethereum
Abrufen von Kursdaten für Wrapped Ethereum…
Fehler beim Abrufen der API für Wrapped Ethereum: 404
Abrufen von Kursdaten für Solana…
Erfolgreich abgerufen: Solana
Abrufen von Kursdaten für Avalanche…
Fehler beim Abrufen der API für Avalanche: 404
Abrufen von Kursdaten für Polkadot…
Erfolgreich abgerufen: Polkadot
Abrufen von Kursdaten für Near Protocol…
API-Limit erreicht für Near Protocol. Warte 60 Sekunden…
```

0.7 7 Analyse und Dashboard mit Streamlit

0.7.1 Einrichtung der Streamlit-App

Streamlit ist ein einfaches Framework zur Erstellung interaktiver Dashboards mit Python. Die folgende Anwendung visualisiert die analysierten Sentiment-Daten.

Installation der benötigten Pakete

```
pip install streamlit pandas gdown matplotlib seaborn
```

Aufbau der Streamlit-App Die folgende **app.py** Datei enthält das vollständige Dashboard für die Visualisierung der analysierten Reddit-Daten:

[ ]:
```
import streamlit as st
import pandas as pd
import gdown
import os
import matplotlib.pyplot as plt
import seaborn as sns

st.set_page_config(page_title="Krypto-Sentiment Dashboard", layout="centered")

MERGED_CSV_ID = "102W-f_u58Jvx9xBAv4IaYrOY6txk-XKL"
MERGED_CSV = "reddit_merged.csv"

@st.cache_data
def download_csv(file_id, output):
    url = f"https://drive.google.com/uc?id={file_id}"
    gdown.download(url, output, quiet=False)

@st.cache_data
def load_data():
    if not os.path.exists(MERGED_CSV):
        download_csv(MERGED_CSV_ID, MERGED_CSV)
    df = pd.read_csv(MERGED_CSV, sep="|", encoding="utf-8-sig",␣
 ↪on_bad_lines="skip")
    df["date"] = pd.to_datetime(df["date"], errors="coerce")
    df["sentiment_score"] = df["sentiment"].map({"positive": 1, "neutral": 0,␣
 ↪"negative": -1})
    return df

df_merged = load_data()

st.title(" Krypto-Sentiment Dashboard")

if df_merged.empty:
    st.warning(" Keine Daten verfügbar. Überprüfe Google Drive oder lade neue␣
 ↪Daten hoch.")
else:
    st.subheader(" Top 10 meist erwähnte Kryptowährungen")
    crypto_counts = df_merged["crypto"].value_counts().head(10)
    st.bar_chart(crypto_counts)

    st.subheader(" Sentiment-Verteilung der Coins")
    sentiment_distribution = df_merged.groupby(["crypto", "sentiment"]).size().
 ↪unstack(fill_value=0)
    st.bar_chart(sentiment_distribution)
```

```
    st.subheader("  Verhältnis Positiv vs. Negativ")
    sentiment_ratio = df_merged[df_merged["sentiment"] != "neutral"].
↪groupby("sentiment").size()
    fig, ax = plt.subplots(figsize=(5, 5))
    ax.pie(sentiment_ratio, labels=sentiment_ratio.index, autopct="%1.1f%%",␣
↪startangle=90, colors=["green", "red"])
    ax.axis("equal")
    st.pyplot(fig)

    st.subheader("  Sentiment-Entwicklung über Zeit")
    crypto_options = df_merged["crypto"].unique().tolist()
    selected_crypto = st.selectbox("Wähle eine Kryptowährung:", crypto_options,␣
↪index=0)
    df_filtered = df_merged[(df_merged["crypto"] == selected_crypto) &␣
↪(df_merged["sentiment"] != "neutral")]
    df_time = df_filtered.groupby(["date", "sentiment"]).size().
↪unstack(fill_value=0)
    st.line_chart(df_time)

st.write("  Dashboard wird regelmäßig mit neuen Daten aktualisiert!")
```

0.7.2 Starten der Streamlit-App

Die App kann mit folgendem Befehl gestartet werden:

```
streamlit run app.py
```

0.7.3 Möglichkeiten zur Erweiterung

Die Anwendung kann um weitere Visualisierungen und Features ergänzt werden:

- Erweiterte Sentiment-Trends: Liniendiagramme für Langzeitanalysen.
- Korrelationen zwischen Coins: Heatmaps zur Analyse von Korrelationen zwischen Kryptowährungen.
- Interaktive Benutzersteuerung: Mehr Auswahlmöglichkeiten für den Nutzer, wie Filter für bestimmte Zeiträume oder Sentiment-Typen.
- Integration weiterer Datenquellen: Kombination mit Twitter-Daten oder Finanzmarktdaten zur besseren Analyse der Marktentwicklung.
- Verbesserte NLP-Modelle: Einsatz von FinBERT, CryptoBERT oder GPT-basierten Modellen, um Sentiment-Analysen genauer zu machen und Fake News oder Spam zu erkennen.
- Prognosemodelle für Preisbewegungen: Nutzung von Regressionsmodellen, LSTMs oder Random Forests, um die Auswirkungen von Sentiment-Trends auf Kursbewegungen vorherzusagen.
- Berechnung der logarithmischen Rendite: Analyse der Log-Rendite von Kryptowährungen als Basis für Vorhersagemodelle und Risikomanagement.

Durch die Nutzung von Matplotlib, Seaborn und Streamlit stehen zahlreiche Möglichkeiten für kreative Datenvisualisierung zur Verfügung. Entwickler können das Dashboard kontinuierlich

verbessern, neue NLP-Modelle integrieren und Predictive Analytics für den Kryptomarkt anwenden.