Name: Deep Salunkhe

Roll No: 21102A0014

Hashtag/Keyword Trend Analysis

1. Setup and Installation

```
!pip install asyncpraw praw pandas matplotlib textblob
```

```
Collecting asyncpraw
  Downloading asyncpraw-7.8.1-py3-none-any.whl.metadata (9.0 kB)
Collecting praw
  Downloading praw-7.8.1-py3-none-any.whl.metadata (9.4 kB)
Requirement already satisfied: pandas in
/usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: textblob in
/usr/local/lib/python3.11/dist-packages (0.19.0)
Collecting aiofiles (from asyncpraw)
  Downloading aiofiles-24.1.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: aiohttp<4 in
/usr/local/lib/python3.11/dist-packages (from asyncpraw) (3.11.15)
Collecting aiosqlite<=0.17.0 (from asyncpraw)
  Downloading aiosqlite-0.17.0-py3-none-any.whl.metadata (4.1 kB)
Collecting asyncprawcore<3,>=2.4 (from asyncpraw)
  Downloading asyncprawcore-2.4.0-py3-none-any.whl.metadata (5.5 kB)
Collecting update_checker>=0.18 (from asyncpraw)
  Downloading update_checker-0.18.0-py3-none-any.whl.metadata (2.3 kB)
Collecting prawcore<3,>=2.4 (from praw)
  Downloading prawcore-2.4.0-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: websocket-client>=0.54.0 in
/usr/local/lib/python3.11/dist-packages (from praw) (1.8.0)
Requirement already satisfied: numpy>=1.23.2 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in
```

```
/usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.3)
Requirement already satisfied: nltk>=3.9 in
/usr/local/lib/python3.11/dist-packages (from textblob) (3.9.1)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp<4->asyncpraw)
(2.6.1)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.11/dist-packages (from aiohttp<4->asyncpraw)
(1.3.2)
Requirement already satisfied: attrs>=17.3.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp<4->asyncpraw)
(25.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.11/dist-packages (from aiohttp<4->asyncpraw)
(1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.11/dist-packages (from aiohttp<4->asyncpraw)
(6.4.3)
Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp<4->asyncpraw)
(0.3.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp<4->asyncpraw)
(1.19.0)
Requirement already satisfied: typing_extensions>=3.7.2 in
/usr/local/lib/python3.11/dist-packages (from aiosqlite<=0.17.0-
>asyncpraw) (4.13.2)
Requirement already satisfied: click in
/usr/local/lib/python3.11/dist-packages (from nltk>=3.9->textblob)
(8.1.8)
Requirement already satisfied: joblib in
/usr/local/lib/python3.11/dist-packages (from nltk>=3.9->textblob)
(1.4.2)
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.11/dist-packages (from nltk>=3.9->textblob)
(2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-
```

```
packages (from nltk>=3.9->textblob) (4.67.1)
Requirement already satisfied: requests<3.0,>=2.6.0 in
/usr/local/lib/python3.11/dist-packages (from prawcore<3,>=2.4->praw)
(2.32.3)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2-
>pandas) (1.17.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests<3.0,>=2.6.0-
>prawcore<3,>=2.4->praw) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.11/dist-packages (from requests<3.0,>=2.6.0-
>prawcore<3,>=2.4->praw) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests<3.0,>=2.6.0-
>prawcore<3,>=2.4->praw) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests<3.0,>=2.6.0-
>prawcore<3,>=2.4->praw) (2025.1.31)
Downloading asyncpraw-7.8.1-py3-none-any.whl (196 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 196.4/196.4 kB 7.9 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 189.3/189.3 kB 17.0 MB/s eta
0:00:00
```

```
import praw
from textblob import TextBlob
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
import numpy as np
```

2. Reddit API Configuration

```
CLIENT_ID = 'wR4s22ZsHO85tg5kvqpx7g'
CLIENT_SECRET = 'Ko7OcgyNlmVjupa-OlDaHbTmCwpURA'
USER_AGENT = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124
Safari/537.36'

reddit = praw.Reddit(
  client_id=CLIENT_ID,
  client_secret=CLIENT_SECRET,
  user_agent=USER_AGENT,
  check_for_async=False
)
```

3. Data Collection Function

```
def collect_mentions(keyword, limit=100, time_filter='month'):
    """
    Collect mentions of a keyword/hashtag from Reddit over the
specified time period

    Args:
        keyword (str): The keyword/hashtag to search for
        limit (int): Maximum number of posts to retrieve
        time_filter (str): Time period to search ('hour', 'day',
'week', 'month', 'year', 'all')

    Returns:
        list: List of dictionaries containing mention data
    """
    mentions = []

    # Search for submissions
    for submission in reddit.subreddit("all").search(keyword,
limit=limit, time_filter=time_filter):
        author_karma = getattr(submission.author, 'link_karma', 0)
        created_date = datetime.fromtimestamp(submission.created_utc)

        mentions.append({
            'type': 'post',
            'text': submission.title + " " + submission.selftext,
            'author': getattr(submission.author, 'name', 'Deleted'),
            'karma': author_karma,
            'upvotes': submission.score,
            'date': created_date,
            'timestamp': submission.created_utc
        })

        # Get some comments from each post
        submission.comments.replace_more(limit=0)
        for comment in submission.comments[:3]:  # Limit to 3 comments
per post
            comment_date = datetime.fromtimestamp(comment.created_utc)
            mentions.append({
                'type': 'comment',
                'text': comment.body,
                'author': getattr(comment.author, 'name', 'Deleted'),
                'karma': getattr(comment.author, 'link_karma', 0),
                'upvotes': comment.score,
                'date': comment_date,
                'timestamp': comment.created_utc
            })

    return mentions
```

4. Sentiment Analysis Function

```
def analyze_sentiment(mentions):
    """
    Perform sentiment analysis on collected mentions

    Args:
        mentions (list): List of mention dictionaries

    Returns:
        DataFrame: Pandas DataFrame with sentiment analysis results
    """
    data = [{
        'type': mention['type'],
        'text': mention['text'],
        'author': mention['author'],
        'karma': mention['karma'],
        'upvotes': mention['upvotes'],
        'date': mention['date'],
        'timestamp': mention['timestamp'],
        'sentiment': TextBlob(mention['text']).sentiment.polarity
    } for mention in mentions]

    return pd.DataFrame(data)
```

5. Trend Analysis Functions

```
def analyze_trends(df, keyword):
    """
    Analyze trends in keyword mentions over time

    Args:
        df (DataFrame): DataFrame containing mention data
        keyword (str): The keyword being analyzed
    """
    # Set up figure
    plt.figure(figsize=(15, 10))

    # Convert timestamp to datetime and set as index
    df['datetime'] = pd.to_datetime(df['timestamp'], unit='s')
    df.set_index('datetime', inplace=True)

    # Resample by day
    daily_counts = df.resample('D').size()

    # Plot daily mentions
    plt.subplot(2, 2, 1)
    daily_counts.plot(title=f'Daily Mentions of "{keyword}"',
color='blue')
    plt.xlabel('Date')
```

```
    plt.ylabel('Number of Mentions')

    # Plot sentiment over time
    plt.subplot(2, 2, 2)
    df['sentiment'].resample('D').mean().plot(title='Daily Average
Sentiment', color='green')
    plt.xlabel('Date')
    plt.ylabel('Sentiment Polarity')

    # Plot upvotes distribution
    plt.subplot(2, 2, 3)
    df['upvotes'].plot(kind='hist', bins=20, title='Distribution of
Upvotes', color='orange')
    plt.xlabel('Upvotes')

    # Plot sentiment distribution
    plt.subplot(2, 2, 4)
    df['sentiment'].plot(kind='hist', bins=20, title='Distribution of
Sentiment', color='purple')
    plt.xlabel('Sentiment Polarity')

    plt.tight_layout()
    plt.show()

    # Print summary statistics
    print(f"\nTrend Analysis Summary for '{keyword}':")
    print(f"Total mentions collected: {len(df)}")
    print(f"Time period covered: {df.index.min().date()} to
{df.index.max().date()}")
    print(f"Average daily mentions: {daily_counts.mean():.1f}")
    print(f"Average sentiment score: {df['sentiment'].mean():.2f}")
    print(f"Median upvotes per mention: {df['upvotes'].median()}")

    # Identify spikes in activity
    spike_threshold = daily_counts.mean() + 2 * daily_counts.std()
    spikes = daily_counts[daily_counts > spike_threshold]

    if not spikes.empty:
        print("\nSignificant spikes in activity detected on:")
        for date, count in spikes.items():
            print(f"- {date.date()}: {count} mentions")
    else:
        print("\nNo significant spikes in activity detected.")

    return daily_counts

def identify_top_contributors(df, top_n=5):
    """
    Identify top contributors based on activity and engagement
```

```
    Args:
        df (DataFrame): DataFrame containing mention data
        top_n (int): Number of top contributors to return

    Returns:
        DataFrame: Top contributors with their stats
    """
    return (df.groupby('author')
            .agg({'karma': 'max', 'upvotes': 'sum', 'sentiment':
'mean', 'type': 'count'})
            .rename(columns={'type': 'mentions'})
            .reset_index()
            .sort_values(by=['upvotes', 'mentions'], ascending=[False,
False])
            .head(top_n))
```

6. Execute Analysis

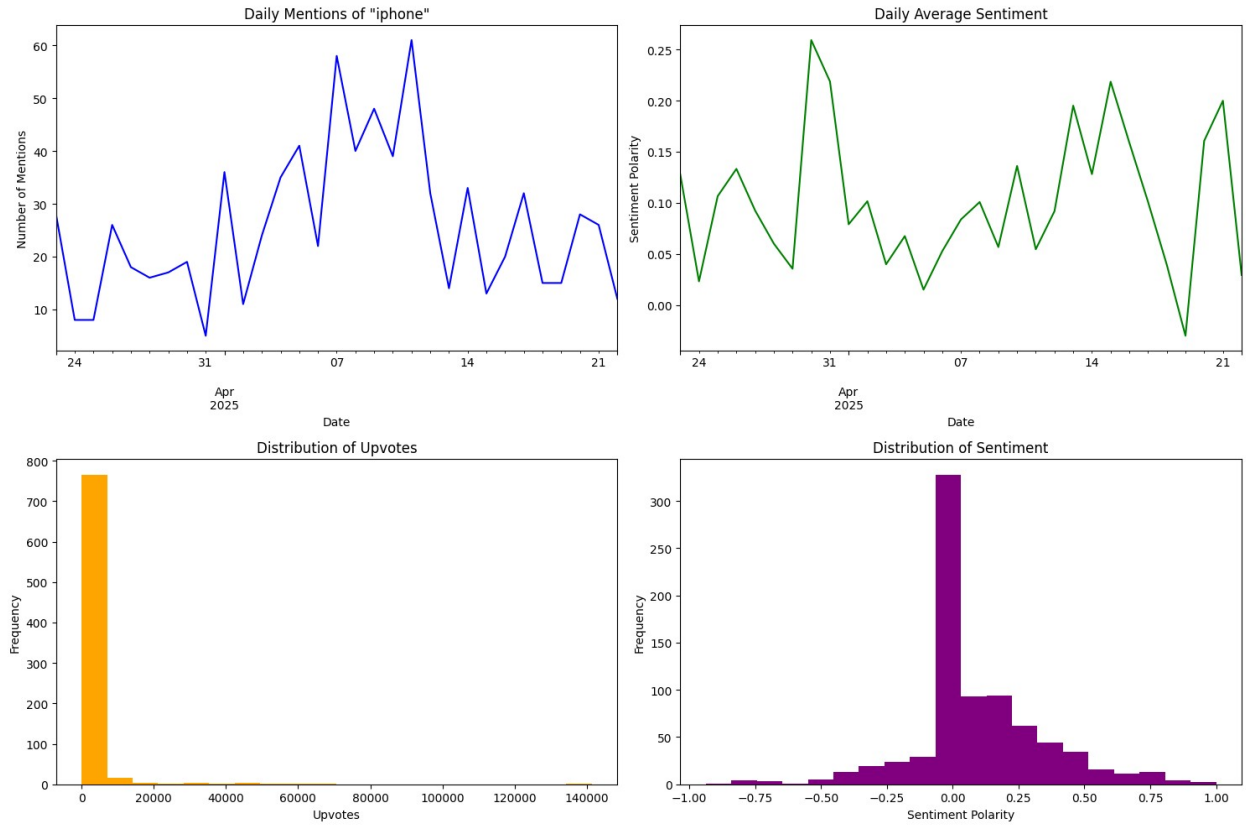
```
# Set the keyword/hashtag to analyze
keyword = "iphone"

# Collect data from the past month
mentions = collect_mentions(keyword, limit=200, time_filter='month')

# Perform sentiment analysis
df = analyze_sentiment(mentions)

# Display the first few rows of data
print("\nSample of collected data:")
display(df.head())
```

```
Sample of collected data:
```


```
{"summary":"{\n  \"name\": \"display(df\",\n  \"rows\": 5,\n
\"fields\": [\n    {\n      \"column\": \"type\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n
\"comment\",\n          \"post\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"text\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 5,\n
\"samples\": [\n          \"Get your new iPhone 17, made 100% in the
USA, only $17,000.\",\n          \"My wife said she\\u2019s nearly
filled a 512GB iPhone and needs a 1TB. This is what I found when I
looked at her storage \"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"author\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          \"Bongeh\",\n          \"ItsGettinBreesy\"\n        ],\n
```

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"karma\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 61804,\n
\"min\": 31,\n   \"max\": 151582,\n   \"num_unique_values\":
5,\n   \"samples\": [\n   746,\n   151582\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"upvotes\",\n   \"properties\": {\n   \"dtype\": \"number\",\
n   \"std\": 17496,\n   \"min\": 1075,\n   \"max\":
43376,\n   \"num_unique_values\": 5,\n   \"samples\": [\n
10458,\n   43376\n   ],\n   \"semantic_type\": \"\",\
n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"date\",\n   \"properties\": {\n   \"dtype\":
\"date\",\n   \"min\": \"2025-03-30 16:56:54\",\n   \"max\":
\"2025-04-09 09:09:56\",\n   \"num_unique_values\": 5,\n
\"samples\": [\n   \"2025-04-09 09:09:56\",\n   \"2025-
03-30 16:56:54\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"timestamp\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 373553.3119382025,\n   \"min\":
1743353814.0,\n   \"max\": 1744189796.0,\n
\"num_unique_values\": 5,\n   \"samples\": [\n
1744189796.0,\n   1743353814.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"sentiment\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0.18939270797263577,\n   \"min\": -0.0879794973544974,\n
\"max\": 0.4,\n   \"num_unique_values\": 4,\n   \"samples\":
[\n   0.06818181818181818,\n   0.4\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe"}

```
# Analyze trends
daily_counts = analyze_trends(df.copy(), keyword)
```




```
Trend Analysis Summary for 'iphone':
Total mentions collected: 800
Time period covered: 2025-03-23 to 2025-04-22
Average daily mentions: 25.8
Average sentiment score: 0.09
Median upvotes per mention: 275.0

Significant spikes in activity detected on:
- 2025-04-07: 58 mentions
- 2025-04-11: 61 mentions

# Identify top contributors
top_contributors = identify_top_contributors(df, top_n=10)
print("\nTop Contributors:")
display(top_contributors)


Top Contributors:

{"summary":"{\n  \"name\": \"top_contributors\",\n  \"rows\": 10,\n
\"fields\": [\n    {\n      \"column\": \"author\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 10,\n        \"samples\": [\n
\"ChickenLuna\",\n          \"WinTraditional4038\",\n
```

```
\"ItsGettinBreesy\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"karma\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 702992,\n        \"min\": 9400,\n        \"max\": 1947143,\n
\"num_unique_values\": 10,\n        \"samples\": [\n          9400,\n
48881,\n          151582\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"upvotes\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 32161,\n        \"min\": 30780,\n
\"max\": 141388,\n        \"num_unique_values\": 10,\n
\"samples\": [\n          31723,\n          65951,\n          43376\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"sentiment\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.2173227246899566,\n        \"min\": -0.5,\n        \"max\": 0.4,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          0.1,\n
-0.075,\n          0.4\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"mentions\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 1,\n
\"max\": 1,\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"top_contributors"}
```

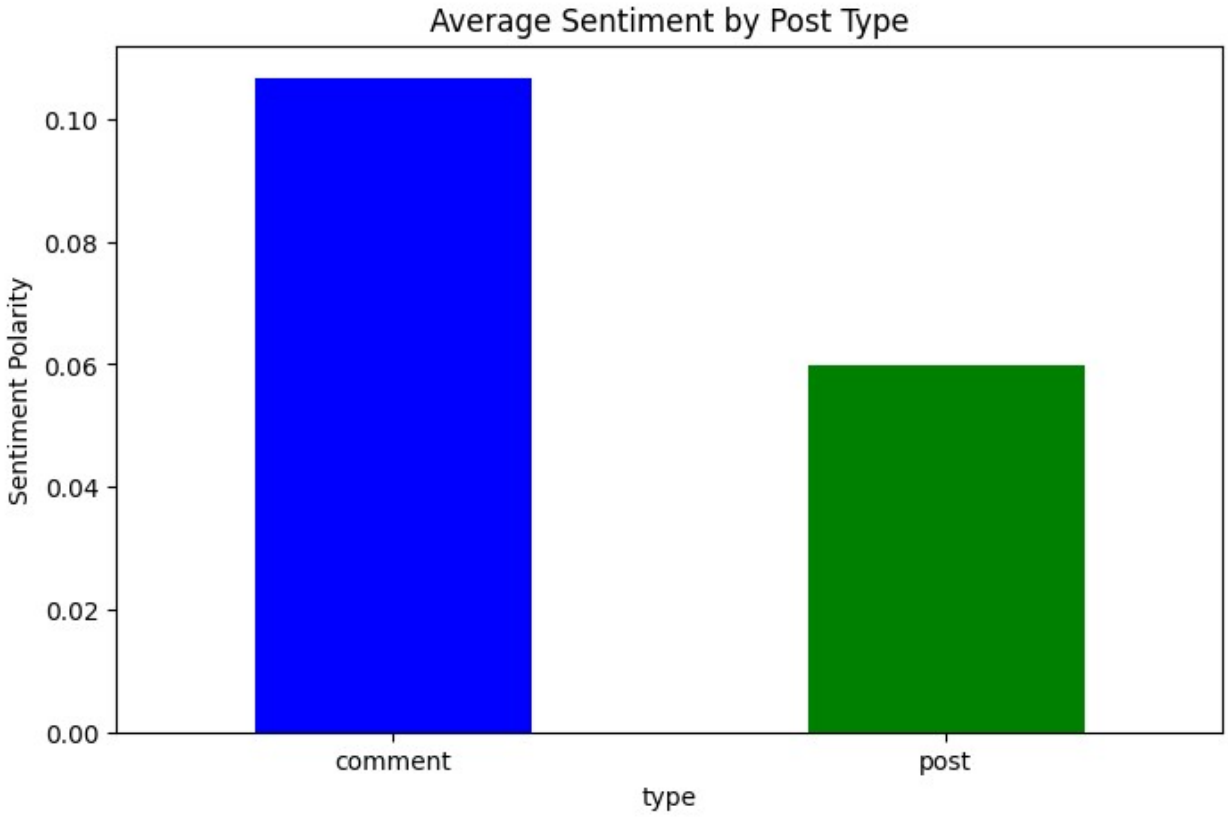

7. Additional Analysis

```
# Sentiment analysis by post type
if 'type' in df.columns:
    print("\nSentiment by Post Type:")
    display(df.groupby('type')['sentiment'].agg(['mean', 'count']))

    # Plot sentiment by type
    plt.figure(figsize=(8, 5))
    df.groupby('type')['sentiment'].mean().plot(kind='bar',
color=['blue', 'green'])
    plt.title('Average Sentiment by Post Type')
    plt.ylabel('Sentiment Polarity')
    plt.xticks(rotation=0)
    plt.show()
```

```
Sentiment by Post Type:
```


```
{"summary":"{\n  \"name\": \"    plt\",\n  \"rows\": 2,\n  \"fields\":
[\n    {\n      \"column\": \"type\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"post\",\n          \"comment\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
```

```
\"mean\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.033063258459094906,\n        \"min\":
0.059865568646463335,\n        \"max\": 0.10662407717556231,\n
\"num_unique_values\": 2,\n        \"samples\": [\n
0.059865568646463335,\n          0.10662407717556231\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"count\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 282,\n
\"min\": 200,\n        \"max\": 600,\n        \"num_unique_values\":
2,\n        \"samples\": [\n          200,\n          600\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```

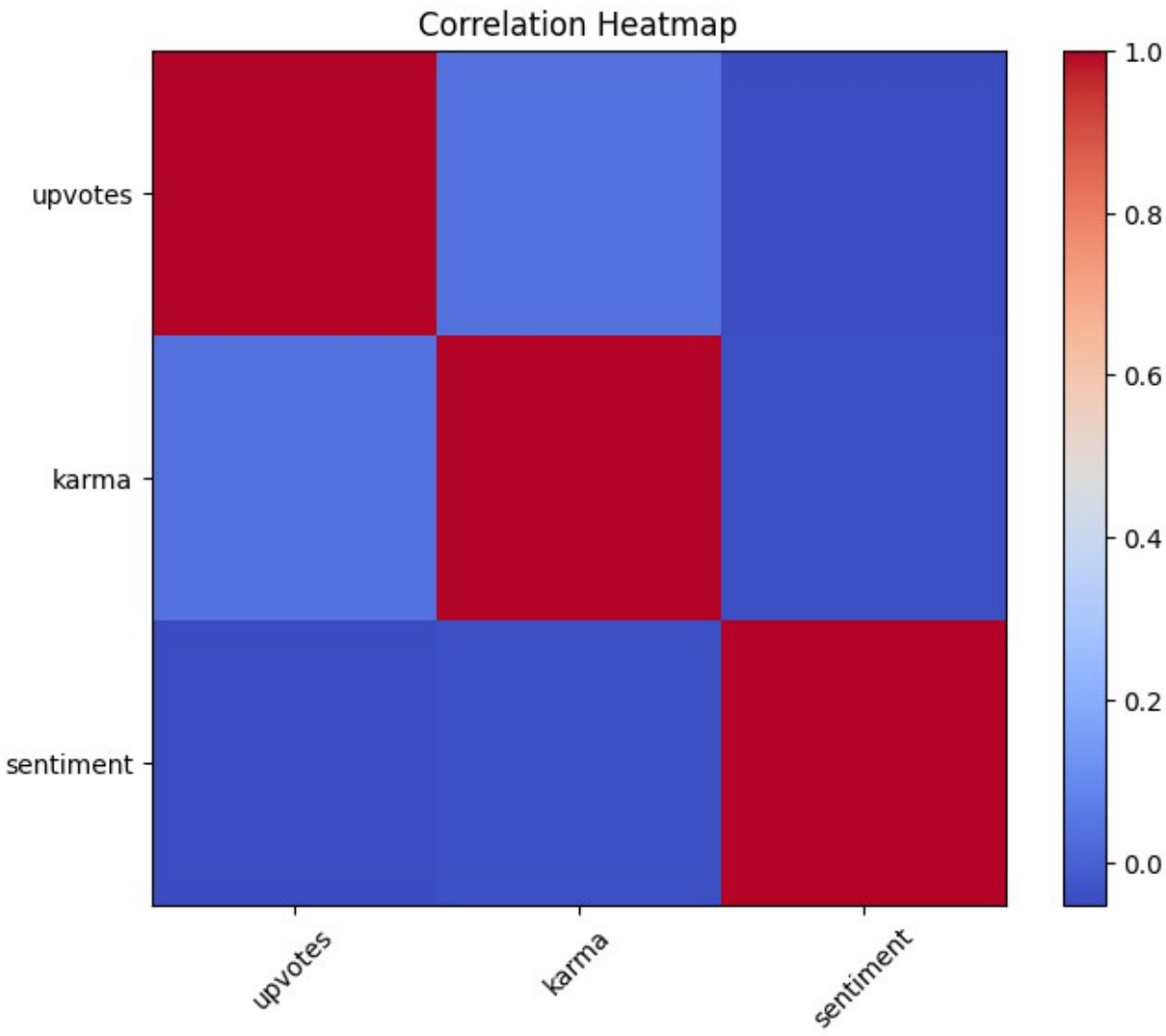




```python
# Correlation analysis
print("\nCorrelation Matrix:")
corr_matrix = df[['upvotes', 'karma', 'sentiment']].corr()
display(corr_matrix)

# Plot correlation heatmap
plt.figure(figsize=(8, 6))
plt.imshow(corr_matrix, cmap='coolwarm', interpolation='none')
plt.colorbar()
plt.xticks(range(len(corr_matrix)), corr_matrix.columns, rotation=45)
plt.yticks(range(len(corr_matrix)), corr_matrix.columns)
```

```
plt.title('Correlation Heatmap')
plt.show()
```

```
Correlation Matrix:
```


```
{"summary":"{\n  \"name\": \"corr_matrix\",\n  \"rows\": 3,\n
\"fields\": [\n    {\n      \"column\": \"upvotes\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.5834552348929399,\n        \"min\": -0.05278904601453291,\n
\"max\": 1.0,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          1.0,\n          0.03773112374883813,\n          -
0.05278904601453291\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"karma\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.5805545556695213,\n        \"min\": -0.04385971183794067,\n
\"max\": 1.0,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          0.03773112374883813,\n          1.0,\n          -
0.04385971183794067\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"sentiment\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.6052668290948789,\n        \"min\": -
0.05278904601453291,\n        \"max\": 1.0,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          -
0.05278904601453291,\n          -0.04385971183794067,\n          1.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"corr_matrix"}
```



8. Save Results

```
# Save dataframe to CSV
df.to_csv(f'{keyword}_mentions_analysis.csv', index=False)
print(f"\nAnalysis results saved to
'{keyword}_mentions_analysis.csv'")
```

```
Analysis results saved to 'iphone_mentions_analysis.csv'
```