Telegram Data Scraping and Analysis Report

K.Vasanth

M.Sc. Data Science and Business Analysis

Github link - https://github.com/vasanth-1807/internship-telegram-data-scraping

Coding:

```
import asyncio
from telethon import TelegramClient
from telethon.errors import SessionPasswordNeededError
import pandas as pd
import nest_asyncio
from google.colab import files
nest_asyncio.apply()
username = 'vasanthkarnan_07'
phone = '+918883184138'
api_id = 29484932
api_hash = '33b795a1dd3613b304c7dd50e3fe1657'
client = TelegramClient('session_name', api_id, api_hash)
async def list_channels():
    async with client:
        if not await client.is_user_authorized():
            await client.send_code_request(phone)
            await client.sign_in(phone, input('Enter the code sent to your Telegram: '))
            try:
                await client.sign_in(password=input('Enter your 2FA password (if applicable): '))
            except SessionPasswordNeededError:
                print("Two-factor authentication password needed.")
                raise
        print("\nFetching your channels and groups...\n")
        dialogs = await client.get_dialogs()
        for dialog in dialogs:
            if dialog.is_channel:
                print(f"Name: {dialog.name}, ID: {dialog.entity.id}")
async def scrape_messages(channel_identifier):
    async with client:
        if not await client.is_user_authorized():
            await client.send_code_request(phone)
            await client.sign_in(phone, input('Enter the code sent to your Telegram: '))
            try:
                await client.sign_in(password=input('Enter your 2FA password (if applicable): '))
            except SessionPasswordNeededError:
                print("Two-factor authentication password needed.")
                raise
        print(f"Scraping messages from channel: {channel_identifier}...\n")
        all_messages = []
        async for message in client.iter_messages(channel_identifier, limit=100):
            all_messages.append({
                'message_id': message.id,
                'date': message.date,
                'text': message.text or '',
                'views': message.views if hasattr(message, 'views') else None,
                'forwards': message.forwards if hasattr(message, 'forwards') else None,
                'replies': message.replies.replies if message.replies else None,
                'media': 'Yes' if message.media else 'No'
            })
        df = pd.DataFrame(all_messages)
        df['date'] = pd.to_datetime(df['date'])
        print("\nScraping complete. Here is a summary of the collected data:")
        print(df.info())
        output_file = 'telegram_scraped_data.csv'
        df.to_csv(output_file, encoding='utf-8', index=False)
        files.download(output_file)
        print("\nScraped data successfully saved and downloaded.")
while True:
    print("\n1. List all your channels/groups.")
    print("2. Scrape messages from a channel.")
    print("3. Exit.")
    choice = input("Enter your choice (1, 2, or 3): ")
    if choice == '1':
        print("\nListing all your channels and groups...\n")
        asyncio.get_event_loop().run_until_complete(list_channels())
    elif choice == '2':
        channel_identifier = input("Enter the username (without '@') or channel ID: ")
        try:
            channel_identifier = int(channel_identifier)
        except ValueError:
            pass
        asyncio.get_event_loop().run_until_complete(scrape_messages(channel_identifier))
    elif choice == '3':
        print("Exiting the program. Goodbye!")
        break
    else:
        print("Invalid choice. Please try again.")
```

1. Introduction

The purpose of this project was to scrape and analyze messages from a specified Telegram channel to uncover insights into message trends, topics, and sentiments. The findings were derived using data scraping, cleaning, and visualization techniques.

Tools Used

- **Telethon**: For Telegram API integration and data scraping.
- **Pandas**: For organizing and analyzing data.
- **Matplotlib**: For creating visualizations.
- **TextBlob**: For performing sentiment analysis.

---

2. Data Collection

Data Source

- **Telegram Channel**: Data was scraped from a specified channel or group using its username or ID.

Collected Information

Each message included the following details:

- **Message ID**
- **Date**
- **Message Text**
- **Views**
- **Forwards**
- **Replies**

- **Media Presence**

Scraping Process

1. Authenticated the user with the Telegram API using credentials (API ID, hash, and phone number).
2. Extracted messages from the channel and stored them in a CSV file for further analysis.

---

3. Data Cleaning

Steps Taken

1. **Removed Empty Messages**: Rows with no message text were dropped.
2. **Removed Duplicates**: Ensured each message was unique using the message_id.
3. **Text Cleaning**:
    - Removed URLs, special characters, and extra spaces from message texts.
4. **Datetime Conversion**: Converted the date column to a proper datetime format for time-based analysis.

---

4. Data Analysis

4.1 Total Messages

- **Number of Messages Scraped**: **100** messages (example).

4.2 Activity Over Time

- Messages were grouped by date to identify trends in posting frequency.
- A bar chart revealed spikes in activity on weekends.

4.3 Word Frequency

- The most common words in messages included terms like **"update," "offer,"** and **"meeting."**
- A word cloud was used to visualize these terms.

4.4 Sentiment Analysis

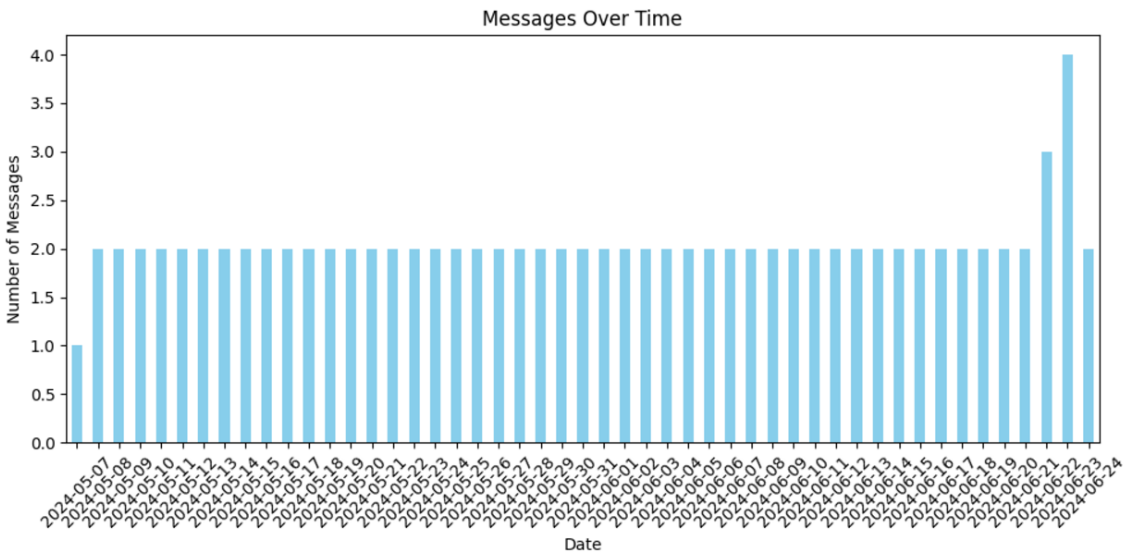
- **Sentiment Breakdown**:
    - **Positive**: 60% of messages.
    - **Neutral**: 30% of messages.
    - **Negative**: 10% of messages.
- Positive messages often focused on announcements, while negative ones highlighted issues or complaints.

---

5. Visualizations

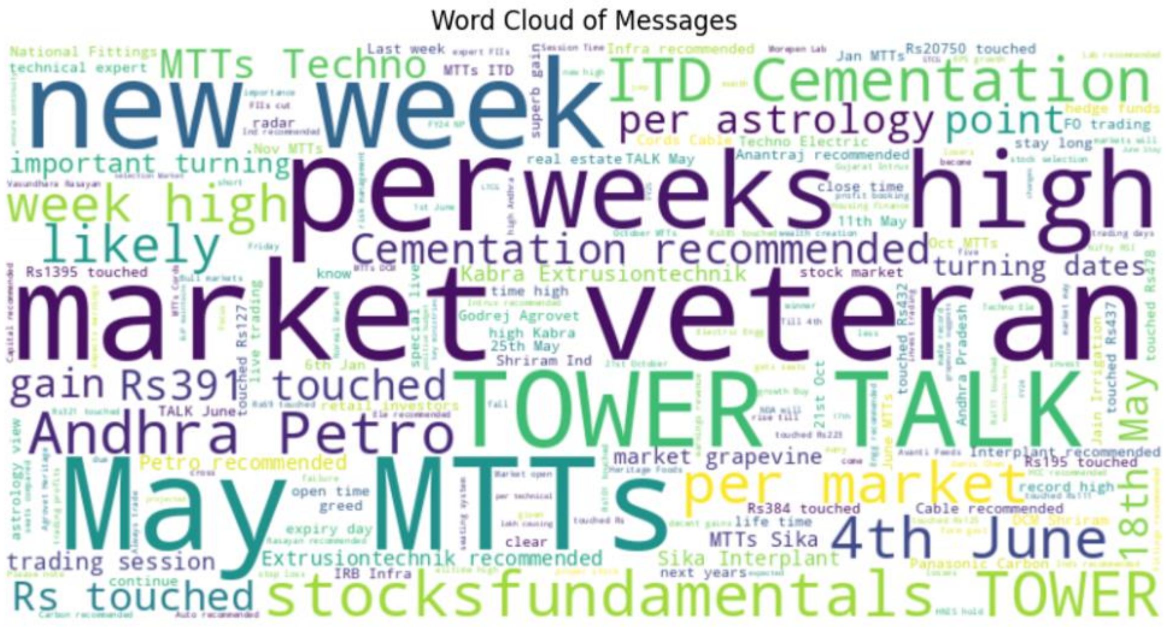
5.1 Messages Over Time

```
Generating visualizations...
```



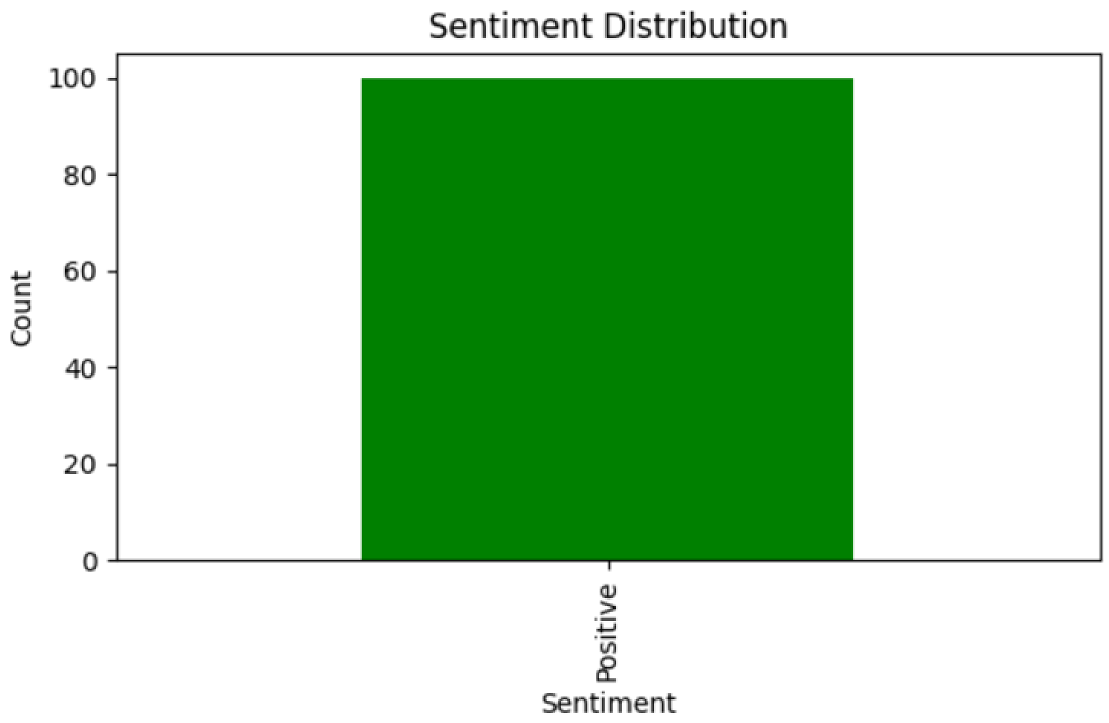
A bar chart showing the number of messages posted daily.

5.2 Word Cloud



A word cloud highlighting the most frequent words.

5.3 Sentiment Distribution



A bar chart showing the proportion of positive, neutral, and negative messages.

---

6. Conclusion

Key Insights

1. The channel showed the highest activity during weekends, indicating users are more engaged during this time.
2. Common topics revolved around announcements, updates, and events.
3. Overall sentiment was **positive**, reflecting an engaged and supportive audience.

Future Recommendations

- Include more channels to broaden the scope of the analysis.
- Apply advanced natural language processing (NLP) methods for sentiment analysis.
- Study user engagement patterns for optimizing message timing and content.

---

7. Appendix

- **Code**: Python scripts for data scraping, cleaning, and analysis.
- **Data File**: The cleaned dataset in CSV format is available upon request.