**Automating Data Collection and Analysis from Telegram Groups using Python and Telethon**

[[Saurabh Chodvadiya](https://medium.com/@chodvadiyasaurabh?source=post_page-----8d51e194fa8b--------------------------------)](https://medium.com/@chodvadiyasaurabh?source=post_page-----8d51e194fa8b--------------------------------)

[Saurabh Chodvadiya](https://medium.com/@chodvadiyasaurabh?source=post_page-----8d51e194fa8b--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F4d5f686a0e1e&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40chodvadiyasaurabh%2Fautomating-data-collection-and-analysis-from-telegram-groups-using-python-and-telethon-8d51e194fa8b&user=Saurabh+Chodvadiya&userId=4d5f686a0e1e&source=post_page-4d5f686a0e1e----8d51e194fa8b---------------------post_header-----------)

4 min read

·

Sep 1

14

Imagine you’re a community manager for a thriving Telegram group, and you want to keep a record of all the messages, usernames, and timestamps for further analysis. Manually copying and pasting messages from the group into a spreadsheet would be a tedious and error-prone task. Here’s where automation comes to the rescue.

**Prerequisites**

Before we dive into the code, make sure you have the following prerequisites in place:

1. Python 3.x installed on your machine.
2. Install the required Python libraries: telethon, pandas, and nltk. You can install them using pip:

In today’s digital age, the wealth of information available on messaging platforms like Telegram can be a goldmine for data analysis. Whether it’s tracking user engagement, sentiment analysis, or simply archiving messages for reference, automating the process of collecting data from Telegram groups can be a game-changer. In this blog, we’ll explore a Python script that leverages the Telethon library to fetch messages from a Telegram group, and we’ll discuss the potential use cases and benefits of such automation.

**Introduction**

Imagine you’re a community manager for a thriving Telegram group, and you want to keep a record of all the messages, usernames, and timestamps for further analysis. Manually copying and pasting messages from the group into a spreadsheet would be a tedious and error-prone task. Here’s where automation comes to the rescue.

**Prerequisites**

Before we dive into the code, make sure you have the following prerequisites in place:

1. Python 3.x installed on your machine.
2. Install the required Python libraries: telethon, pandas, and nltk. You can install them using pip:

pip install telethon pandas nltk

**Setting Up the Telegram API**

To use the Telethon library, you’ll need to set up your Telegram API credentials. Follow these steps:

1. Go to the [Telegram API website](https://my.telegram.org/auth) and log in with your Telegram account.
2. Once logged in, click on the “API development tools” link.
3. Create a new application by filling out the form. You’ll receive an API ID and API hash. Replace the placeholders in the script with your actual API ID and API hash.

**Understanding the Code**

The provided Python script connects to your Telegram account, authenticates, and fetches messages from a specified group within a time range. Let’s break down the code:

import asyncio  
from datetime import datetime, timedelta  
from telethon import TelegramClient, events  
import pandas as pd  
import nltk  
  
# Replace the values with your own API ID, API hash, and phone number  
api\_id = ""  
api\_hash = ''  
phone\_number = '+91'  
  
group\_name = -1001630288049 # Replace with your group ID  
  
start\_time = datetime.now() - timedelta(hours=24)  
flag = 0

In this section, we import the necessary libraries and set up variables such as api\_id, api\_hash, phone\_number, and group\_name.

async def get\_group\_messages():  
 df = pd.DataFrame({'Data': [''], 'name': [''], 'mobile': ['']})  
 df1 = pd.DataFrame({'Data': [''], 'name': [''], 'mobile': ['']})  
 client = TelegramClient('session\_name', api\_id, api\_hash)  
 await client.connect()  
  
 if not await client.is\_user\_authorized():  
 await client.send\_code\_request(phone\_number)  
 await client.sign\_in(phone\_number, input('Enter the code: '))

In this part, we define an asynchronous function get\_group\_messages() that sets up data storage using Pandas DataFrames, connects to the Telegram client, and authorizes the user if necessary.

group = await client.get\_entity(group\_name)  
 date\_today = datetime.utcnow().replace(hour=0, minute=0, second=0, microsecond=0)  
   
 yesterday = date\_today - timedelta(days=5)  
 messages = []

Here, we use the await client.get\_entity(group\_name) method to get the group entity based on its ID. We also set up a time range for message retrieval, in this case, messages from the last 5 days.

async for message in client.iter\_messages(group, min\_id=1):  
 print(message.date, yesterday)  
 if str(message.date) < str(yesterday):  
 break  
 messages.append(message)

This section iterates through messages in the group using client.iter\_messages(). It stops fetching messages when the timestamp of a message is earlier than the specified yesterday. Messages are stored in the messages list.

asyncio.run(get\_group\_messages())

Finally, we execute the get\_group\_messages() function using asyncio.

**Potential Use Cases**

1. Data Analysis: Collecting messages from Telegram groups can be invaluable for sentiment analysis, content trends, and user engagement studies.
2. Moderation: Automating message retrieval can aid in monitoring and moderating group content efficiently.
3. Archiving: Storing messages in a structured format allows you to build archives for reference or historical analysis.
4. Reporting: Generate reports on group activity, such as message frequency, user participation, or keyword analysis.

**Code**

import asyncio  
from datetime import datetime, timedelta  
from telethon import TelegramClient, events  
import pandas as pd  
import nltk  
# Replace the values with your own API ID, API hash and phone number  
# enter your api id from telegram website  
api\_id = ""  
api\_hash = ''  
# enter your api hash form telegram api website  
phone\_number = '+91'  
# enter your pone number on this formate  
  
group\_name = -1001630288049  
# enter your channel group id -100 after this digit  
  
# Set the time range to get messages from  
start\_time = datetime.now() - timedelta(hours=24)  
flag=0  
async def get\_group\_messages():  
 df = pd.DataFrame({'Data':[''],'name':[''],'mobile':['']})  
 df1 = pd.DataFrame({'Data':[''],'name':[''],'mobile':['']})  
 # Create a Telegram client with the specified API ID, API hash and phone number  
 client = TelegramClient('session\_name', api\_id, api\_hash)  
 await client.connect()  
  
 # Check if the user is already authorized, otherwise prompt the user to authorize the client  
 if not await client.is\_user\_authorized():  
 await client.send\_code\_request(phone\_number)  
 await client.sign\_in(phone\_number, input('Enter the code: '))  
  
 # Get the ID of the specified group  
 group = await client.get\_entity(group\_name)  
 date\_today = datetime.utcnow().replace(hour=0, minute=0, second=0, microsecond=0)  
   
 yesterday = date\_today - timedelta(days=5)  
 messages = []  
 # below commented code is used for specified time range  
 async for message in client.iter\_messages(group, min\_id=1):  
 print(message.date ,yesterday)  
 if str(message.date) < str(yesterday):  
 break  
 messages.append(message)  
asyncio.run(get\_group\_messages())

**Conclusion**

Automating data collection from Telegram groups using Python and Telethon can significantly streamline your data analysis workflow. By leveraging the power of asynchronous programming and data manipulation libraries like Pandas, you can efficiently collect, store, and analyze messaging data. Whether you’re a community manager, a data scientist, or simply looking to gain insights from your Telegram group, this script is a valuable tool in your arsenal. With the data collected, you can unlock a wealth of information and make data-driven decisions to enhance your group’s engagement and content strategy.