MEDAC
Instituto Oficial de Formación Profesional

Apis y Bot de Telegram

Youssef Ezzamarty

Índice

Manual de uso

- **/start** para empezar.
- **/help** en caso de necesitar ayuda
- Escribe un mensaje describiendo tu estado de ánimo. El bot entiende el lenguaje natural en varios idiomas.

Funcionamiento de bot.php

1. Importa las dependencias necesarias usando require 'vendor/autoload.php' y use GuzzleHttp\Client para hacer solicitudes HTTP a la API de Telegram y la API de TMDb.
2. Define el token del bot de Telegram y la clave de API de TMDb.
3. Define una función llamada **enviarMensaje** que utiliza GuzzleHttp para enviar mensajes al chat de Telegram.
4. Define una función llamada **manejarMensajeNormal** que procesa el estado de ánimo del usuario, ejecuta un script de Python para analizar el estado de ánimo y luego utiliza la API de TMDb para obtener películas relacionadas.
5. Inicializa el cliente Guzzle HTTP.
6. Utiliza un bucle de polling para obtener actualizaciones de Telegram continuamente. Por cada actualización, verifica si es un mensaje nuevo y si es así, maneja el mensaje normalmente o envía un mensaje de bienvenida si es el comando **/start**.
7. Actualiza el ID del último mensaje procesado para evitar procesar mensajes antiguos repetidamente.
8. Espera un segundo antes de la próxima consulta para evitar un exceso de consumo de recursos.

Procesamiento del estado de ánimo

Uno de los atributos de la API de películas es ***genre_id,*** algunos de los géneros más comunes y sus identificadores en TMDb son:

Acción: 28

Aventura: 12

Animación: 16

Comedia: 35

Crimen: 80

Documental: 99

Drama: 18

Familia: 10751

Fantasía: 14

Historia: 36

Terror: 27

Misterio: 9648

Romance: 10749

Ciencia ficción: 878

Televisión: 10770

Suspense: 53

Según el estado de ánimo seleccionamos algunos ***genre_id*** u otros.

```php
// Obtener el estado de ánimo del mensaje
$estado_animo = $mensaje['text'];

// Ejecutar el script de Python con el estado de ánimo como argumento
$output = exec("python sentiments3.py \"$estado_animo\"");

// Definimos los géneros deseados
$genre_ids_array = [];

// Switch para seleccionar los géneros según el estado de ánimo
switch ($output) {
    case "feliz":
        $genre_ids_array = [28, 12, 16, 35];
        break;
    case "triste":
        $genre_ids_array = [10752, 18, 80, 10751];
        break;
    case "neutro":
        $genre_ids_array = [878, 12, 28, 27];
        break;
    case "contento":
        $genre_ids_array = [28, 12, 878];
        break;
    case "emocionado":
        $genre_ids_array = [18, 27, 53];
        break;
    case "entusiasmado":
        $genre_ids_array = [35, 10751, 10749];
        break;
    case "ansioso":
        $genre_ids_array = [53, 9648, 10749];
        break;
    default:
        enviarMensaje($mensaje['chat']['id'], "Estado de ánimo no reconocido");
        exit;
}
```

En el siguiente punto abordaremos cómo obtener el estado de ánimo.

Procesamiento de lenguaje natural

sentiments.py Usando `textblob`

```python
import sys
from googletrans import Translator
from textblob import TextBlob

# Recibimos el texto desde PHP
texto_usuario = ' '.join(sys.argv[1:])  # Unimos todos los argumentos en una sola cadena

try:
    # Creamos un objeto Translator de Google
    translator = Translator()

    # Traducimos el texto al inglés
    translated_text = translator.translate(texto_usuario, dest='en').text

    # Creamos un nuevo objeto TextBlob con el texto traducido
    translated_blob = TextBlob(translated_text)

    # Obtenemos el sentimiento del texto traducido
    sentimiento = translated_blob.sentiment

    # Convertimos el sentimiento detectado en un estado de ánimo
    e_animo = ""
    if sentimiento.polarity > 0.3:
        e_animo = "feliz"
    elif sentimiento.polarity < -0.3:
        e_animo = "triste"
    elif 0 <= sentimiento.polarity <= 0.3:
        e_animo = "neutro"
    elif 0.3 < sentimiento.polarity <= 0.5:
        e_animo = "contento"
    elif 0.5 < sentimiento.polarity <= 0.7:
        e_animo = "emocionado"
    elif 0.7 < sentimiento.polarity <= 0.9:
        e_animo = "entusiasmado"
    else:
        e_animo = "ansioso"

    # Devolvemos el estado de ánimo al script PHP
    print(e_animo)

except Exception as e:
    print("Error:", e)

```

Ventajas: Funciona correctamente.

Desventajas: Es poco preciso, menos preciso que los demás.

Usando un modelo previamente entrenado.

```python
import sys
from googletrans import Translator
from transformers import pipeline

# Recibimos el texto desde PHP
texto_usuario = ' '.join(sys.argv[1:])  # Unimos todos los argumentos en una sola cadena

try:
    # Creamos un objeto Translator de Google
    translator = Translator()

    # Traducimos el texto al inglés
    translated_text = translator.translate(texto_usuario, dest='en').text

    # Cargamos un modelo pre-entrenado para análisis de sentimientos
    sentiment_analysis = pipeline("sentiment-analysis")

    # Obtenemos el análisis de sentimientos del texto traducido
    sentiment_result = sentiment_analysis(translated_text)[0]

    # Extraemos la etiqueta de sentimiento y la probabilidad
    label = sentiment_result['label']
    score = sentiment_result['score']

    # Convertimos el sentimiento detectado en un estado de ánimo
    if label == 'POSITIVE':
        if score > 0.7:
            e_animo = "emocionado"
        elif score > 0.5:
            e_animo = "entusiasmado"
        elif score > 0.3:
            e_animo = "contento"
        else:
            e_animo = "feliz"
    elif label == 'NEGATIVE':
        e_animo = "triste"
    else:
        e_animo = "neutro"

    # Devolvemos el estado de ánimo al script PHP
    print(e_animo)

except Exception as e:
    print("Error:", e)

```

Ventajas: Más preciso y personalizable.

Desventajas: Muy lento en comparación con los demás. Más difícil de gestionar, a veces devuelve lo que no debería.

Usando `vaderSentiment.vaderSentiment`

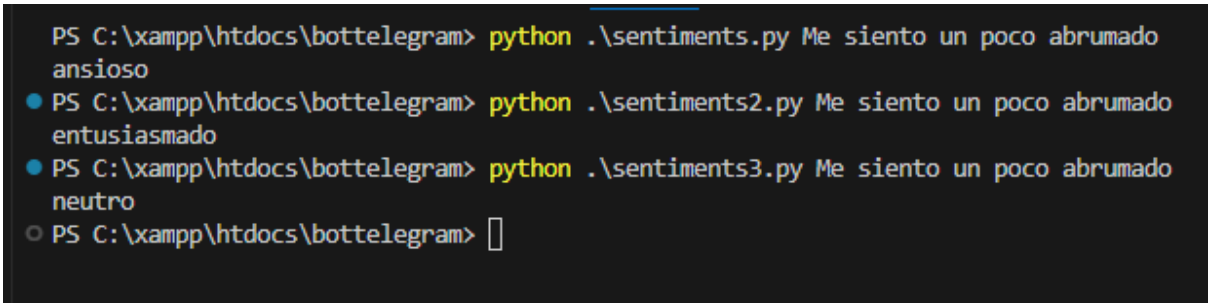
```python
import sys
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from googletrans import Translator

# Recibimos el texto desde PHP
texto_usuario = ' '.join(sys.argv[1:])  # Unimos todos los argumentos en una sola cadena

try:
    # Creamos un objeto Translator de Google
    translator = Translator()

    # Traducimos el texto al inglés
    translated_text = translator.translate(texto_usuario, dest='en').text

    # Creamos un analizador de sentimientos de VADER
    analyzer = SentimentIntensityAnalyzer()

    # Obtenemos el análisis de sentimientos del texto traducido
    sentiment_scores = analyzer.polarity_scores(translated_text)

    # Convertimos el sentimiento detectado en un estado de ánimo
    compound_score = sentiment_scores['compound']

    if compound_score > 0.7:
        e_animo = "entusiasmado"
    elif 0.5 <= compound_score <= 0.7:
        e_animo = "emocionado"
    elif 0.3 <= compound_score < 0.5:
        e_animo = "contento"
    elif -0.3 <= compound_score <= 0.3:
        e_animo = "neutro"
    elif -0.5 <= compound_score < -0.3:
        e_animo = "triste"
    elif compound_score < -0.7:
        e_animo = "ansioso"
    else:
        e_animo = "feliz"

    # Devolvemos el estado de ánimo al script PHP
    print(e_animo)

except Exception as e:
    print("Error:", e)

```

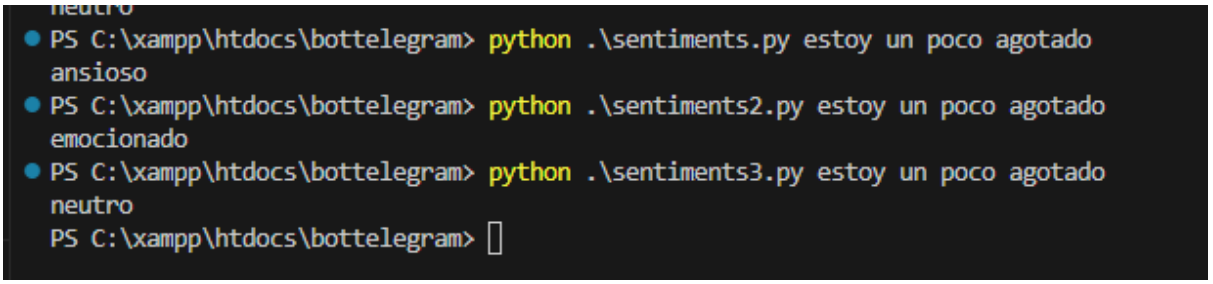
Ventajas: Funciona correctamente.

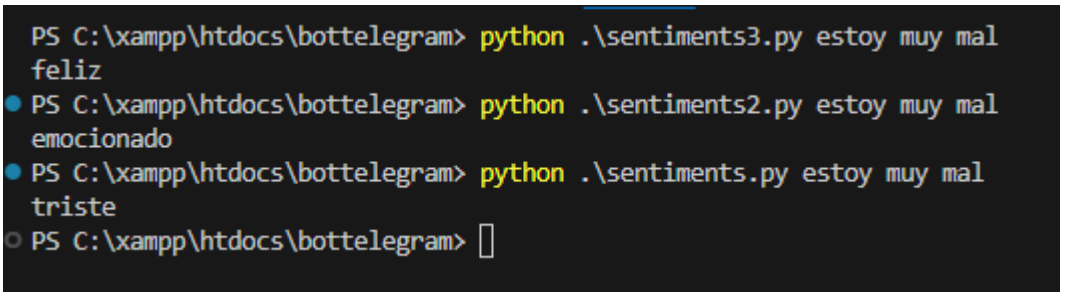
Desventajas: Al principio parecía que no funcionaba correctamente, sin embargo, ajustando el *switch case* da muy buenos resultados .

Comparaciones

Aquí se ejecutan los distintos scripts para ir comparando los tres programas de python y ajustando los algoritmos para obtener resultados adecuados. Estas bibliotecas de NLP (Natural Language Processing) de Python usan un *score* para determinar la polaridad de un mensaje. Según esta polaridad podemos saber si un texto es considerado que expresa un mensaje más positivo o negativo.



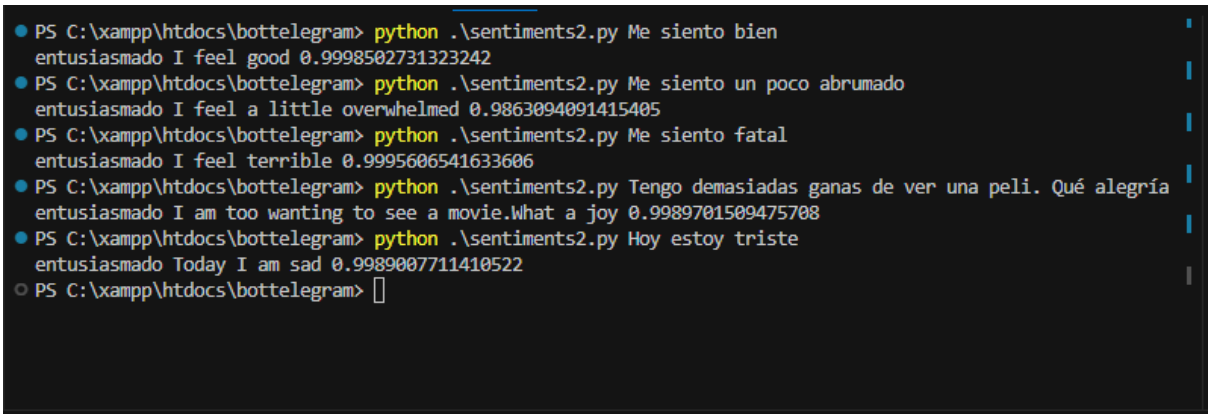




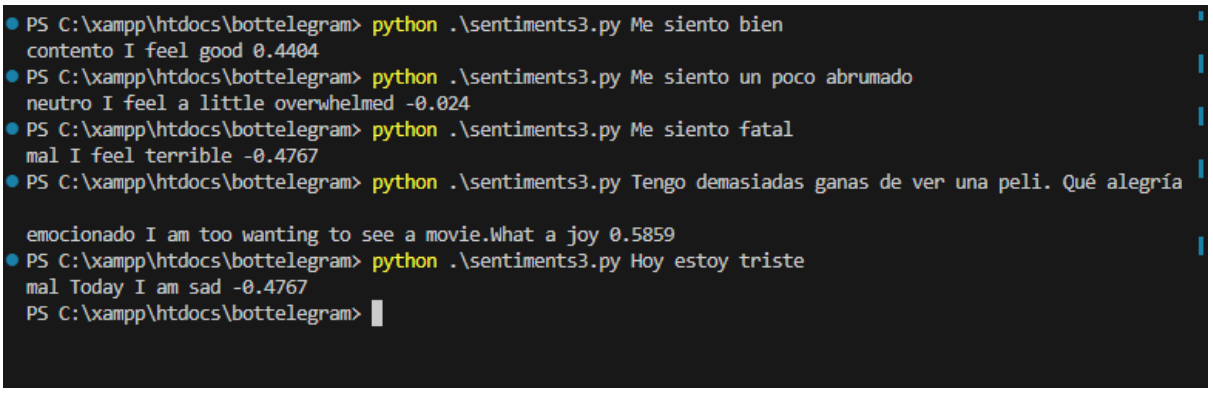
Comparación final

Debido a la falta de precisión del primer script, optamos por comparar sentiments2.py y sentiments3.py después de haber realizado algunos ajustes previamente.

Con modelo entrenado previamente:



Con la biblioteca python: `vaderSentiment.vaderSentiment`



Finalmente ya podemos decidir qué script usaremos. En este caso el ganador parece el tercer código, que emplea `vaderSentiment.vaderSentiment.`

Algunos problemas que vamos a afrontar

- **Dependencias de Python:** Nos podemos encontrar con problemas a la hora de ejecutar los scripts de python, sobre todo relacionado a la biblioteca urlib3.
- **Caché:** Al ejecutar el bot este puede recibir una gran cantidad de mensajes y por lo tanto, devolver una gran cantidad de respuestas de forma descontrolada. Estos mensajes están guardados en la siguiente dirección. Estos se conocen como updates y sirven para ir actualizando el bot en tiempo real.

https://api.telegram.org/bot6554813207:AAGGK4LXVNOr7CV6JM_EeOsdybj_fuXsmzI/getUpdates

```
{
    "ok": true,
    "result": [
        {
            "update_id": 724570444,
            "message": {
                "message_id": 138,
                "from": {
                    "id": 5283220867,
                    "is_bot": false,
                    "first_name": "Sousou",
                    "language_code": "en"
                },
                "chat": {
                    "id": 5283220867,
                    "first_name": "Sousou",
                    "type": "private"
                },
                "date": 1707491567,
                "text": "/start",
                "entities": [
                    {
                        "offset": 0,
                        "length": 6,
                        "type": "bot_command"
                    }
                ]
            }
        },
        {
            "update_id": 724570445,
            "message": {
                "message_id": 140,
                "from": {
                    "id": 5283220867,
                    "is_bot": false,
                    "first_name": "Sousou",
                    "language_code": "en"
                },
                "chat": {
                    "id": 5283220867,
                    "first_name": "Sousou",
                    "type": "private"
                },
                "date": 1707491573,
                "text": "me siento mal"
            }
        }
    ]
}
```


Podemos hacer un script que reciba el JSON de getUpdate mediante el uso de la biblioteca `requests`,que hace peticiones GET de HTTP. A continuación extraemos el campo 'upadte_id' y para finalizar usando la biblioteca mencionada anteriormente hacemos una petición a

https://api.telegram.org/bot6554813207:AAGGK4LXVNOr7CV6JM_EeOsdybj_fuXsmzI/getupdates?offset=[número_ID]+1

Lo ejecutaremos antes de iniciar nuestro bot.

clearUpdates.py > ...

```python
import requests

# URL base
base_url = "https://api.telegram.org/bot6554813207:AAGGK4LXVNOr7CV6JM_EeOsdybj_fuXsmzI/"
#GetUpdates
def get_updates():
    url = base_url + "getUpdates"
    response = requests.get(url)
    if response.status_code == 200:
        return response.json()
    else:
        print("Error:", response.status_code)
        return None
updates = get_updates()
#Ver los update_id
def get_update_ids():
    url = base_url + "getUpdates"
    response = requests.get(url)
    if response.status_code == 200:
        updates = response.json()['result']
        update_ids = [update['update_id'] for update in updates]
        return update_ids
    else:
        print("Error:", response.status_code)
        return None

def get_updates_with_offset(update_ids):
    for update_id in update_ids:
        url = base_url + f"getUpdates?offset={update_id + 1}"
        # Send the GET request
        response = requests.get(url)
        if response.status_code == 200:
            updates = response.json()['result']
            print(updates)
        else:
            print("Error:", response.status_code)
# Call the function to get update IDs
update_ids = get_update_ids()
# Call the function to get updates with offset for each update ID
if update_ids is not None:
    get_updates_with_offset(update_ids)
```

Paso del bot a Python

Debido a problemas de dependencias a la hora de usar PHP he preferido pasar a Python con un script que hace exactamente lo mismo. Además conseguimos un código más corto y legible, a su vez, seremos más rápidos a la hora de trabajar debido a que las bibliotecas son muy accesibles y fáciles de instalar.