SPOTMEL:

Spotify y Mercado Libre en un solo lugar

Informe sobre Trabajo Final Integrador

Realizado por: Azrak Andrés, Carbone Carola, Dolciotti Agustina, Silva Geremías.

Materia: Programación II
Fecha de entrega: 28/11/2024

RESUMEN DEL PROYECTO

El presente proyecto propone resolver la siguiente problemática: cómo integrar en una plataforma de conciertos una asistencia vía chatbot para que el usuario consiga merch sobre el artista que asistirá a ver y a su vez permanezca estratégicamente en dicha plataforma. Esto último se refiere al desafío de desarrollar una estrategia de retención con el fin de aumentar el tiempo y dinero que el usuario invierte en la web. De esta forma, el bot Wanda asistirá al usuario proporcionándole información y el merch sobre el artista que ingrese redirigiéndolo hacia Mercado Libre. Una vez satisfecho el usuario con la búsqueda en esta última plataforma, Wanda le brindará un artista relacionado al ingresado en primera instancia y a su vez la opción de dirigirse hacia su merch también en Mercado Libre. Por ende, el usuario no solamente obtiene merchandising oficial de su artista favorito para su próximo recital, sino que también de un artista relacionado a él. De esta manera, el usuario es retenido estratégicamente a través del circuito diseñado dentro del chatbot para que permanezca la mayor cantidad de tiempo posible en la web y a su vez destina su dinero en elementos para el look de su próximo recital.

OBJETIVOS

1. Desarrollar un código en python que permita aumentar el tiempo y dinero que el usuario invierte en la web.
2. Garantizar la venta del merchandising a través del bot.
3. Incentivar al público a ser creativo e interactuar previamente con el show.
4. Integrar inteligencia artificial para complejizar y perfeccionar el código.

LA TECNOLOGÍA SELECCIONADA

Para ejecutar y llevar a cabo el programa utilizamos un conjunto de tecnologías que consideramos apropiadas y accesibles a la hora de programar.

En primer lugar, decidimos realizar el proyecto con APIs, en este caso la de Mercado Libre y la de Spotify, ya que necesitábamos lograr que a través del código el usuario pueda obtener:

- Una lista de artistas relacionados a otro que ingresó primero. Por esto utilizamos la API de Spotify.

- Una lista de productos relacionados a dichos artistas. Por esto utilizamos la API de Mercado Libre.

Es por estas razones las APIs de Spotify y de Mercado Libre fueron seleccionadas: la forma en que está estructurado su código (simple, compacto e interactivo) y su accesibilidad (gratuitas) resultaron la mejor opción para el proyecto.

En segunda instancia, el código debía a su vez plasmar una breve información acerca del artista que ingresa el usuario bajo el comando "/artist". Dicha información contiene datos tanto biográficos acerca de su estética y estilo. Para lograr esto, debíamos implementar una tecnología con inteligencia artificial, capaz de recolectar información e integrarla en un mismo portal y además interpretar cuál es el estilo de un determinado artista, concepto abstracto que sólo a través de la IA se podría obtener. La tecnología en cuestión es Gemini, utilizada y creada por Google, pues se presentó como la mejor opción ya que además de ser de la que más conocimiento tenemos, es de fácil acceso, rápido procesamiento y está dentro de las más prestigiosas. Es por esto que decidimos implementar la API de Gemini y así perfeccionar el código.

En tercer y último lugar, la programación y ejecución del código per sé fue realizada en Replit. La elección de esta última tecnología se debe a una cuestión práctica, pues contamos con un amplio conocimiento en la misma, es de fácil acceso (gratuita) y además cuenta entre sus servicios con la integración de inteligencia artificial. Esta última herramienta fue de gran ayuda a la hora de programar para los casos en los que surgía algún error en el código.

SOLUCIÓN

A continuación desglosaremos la resolución del programa y el detalle del código desarrollado.

1. **Configuración e inicialización del bot y las APIs**

   En esta sección, se configuran las claves necesarias para interactuar con la APIs de Generative AI, Spotify y Telegram

```
# Configuración de las claves API
os.environ["API_KEY"] = "AIzaSyC1uhmOt89ntEY7trNeAUbTAtNkwcMXbJg"  # Reemplaza con tu API Key de Google
genai.configure(api_key=os.environ["API_KEY"])

# Token del bot de Telegram
TOKEN = '7571232903:AAF6_CnQviBkr4GWjMcDKxE0w6qa9RvajGA'
bot = telebot.TeleBot(TOKEN)

# Configuración de Spotify
client_id = '8fe819ee033e4a6a86ebac251c4245c4'
client_secret = '1f53f4f2543b459a90869e3a1cae633a'
```

2. Interacción con la API de spotify

Utilizamos las funciones de Spotify para conseguir el token de acceso, el ID del artista, la información del artista y sus artistas relacionados.

```
19    # Función para obtener el token de acceso de Spotify
20    def get_access_token(client_id, client_secret):
21        token_url = "https://accounts.spotify.com/api/token"
22        headers = {"Content-Type": "application/x-www-form-urlencoded"}
23        data = {
24            "grant_type": "client_credentials",
25            "client_id": client_id,
26            "client_secret": client_secret
27        }
28        response = requests.post(token_url, headers=headers, data=data)
29        if response.status_code == 200:
30            return response.json()['access_token']
31        else:
32            raise Exception(f"Error al obtener el token: {response.status_code} - {response.text}")
33
```

```
34    # Función para obtener el ID de un artista en Spotify
35    def get_artist_id(access_token, artist_name):
36        search_url = f"https://api.spotify.com/v1/search?q={artist_name}&type=artist"
37        headers = {"Authorization": f"Bearer {access_token}"}
38        response = requests.get(search_url, headers=headers)
39        if response.status_code == 200:
40            artists = response.json().get('artists', {}).get('items', [])
41            if artists:
42                return artists[0]['id']
43            else:
44                raise Exception("Artista no encontrado.")
45        else:
46            raise Exception(f"Error al buscar el artista: {response.status_code} - {response.text}")
```

```python
# Función para obtener artistas relacionados
def get_related_artists(access_token, artist_name):
    artist_id = get_artist_id(access_token, artist_name)
    url = f"https://api.spotify.com/v1/artists/{artist_id}/related-artists"
    headers = {"Authorization": f"Bearer {access_token}"}
    response = requests.get(url, headers=headers)
    if response.status_code == 200:
        return response.json()
    else:
        raise Exception(f"Error al obtener artistas relacionados: {response.status_code} - {response.text}")
```

3. Interacción con la API de Mercado Libre

La función get_merch_from_mercadolibre(artist_name) nos permite simular la búsqueda del merchandising del artista en mercado libre. Este código genera un URL que se utiliza para redirigir al usuario a estos artículos en Mercado Libre

```python
# Función para obtener merchandising de un artista desde Mercado Libre (simulación)
def get_merch_from_mercadolibre(artist_name):
    # Se hace una búsqueda básica en Mercado Libre por el nombre del artista
    # Este URL es un ejemplo. Dependiendo de cómo quieras buscar, ajusta la URL
    search_url = f"https://listado.mercadolibre.com.ar/{artist_name.replace(' ', '-').lower()}-merchandising"
    return search_url

```

4. Generación de contenido con Google AI

Utilizando la función show_artist_info, se utiliza el modelo de IA de google para generar una descripción detallada sobre el artista en base a la consulta realizada.

```python
        # Utilizar el modelo de IA para obtener más detalles sobre el artista
        pregunta = f"detallame muy breve y ordenada, información del artista, su estilo y estetica {artist_name}."
        respuesta = genai.GenerativeModel('gemini-1.5-flash-latest').generate_content(pregunta)

        if respuesta and hasattr(respuesta, 'text'):
            artist_message += f"**Descripción:** {respuesta.text}"

        # Enviar la información del artista al usuario
        bot.send_message(message.chat.id, artist_message)

        # Obtener el merchandising del artista
        merch_url = get_merch_from_mercadolibre(artist_name)
        bot.send_message(message.chat.id, f"¡Aquí está el merchandising disponible para {artist_name} en Mercado Libre! \n{merch_url}")

```

5. Funciones del bot y comandos de Telegram

Configuramos el controlador para el comando /artist y /start los cuales permiten al bot interactuar con el usuario. Le damos inicio el bot con la función polling()

```python
# Comando de bienvenida
@bot.message_handler(commands=['start'])
def send_welcome(message):
    bot.send_message(message.chat.id, "¡Bienvenido/a! Soy Wanda de Spotmel y te ayudo a que encuentres merch para tu próximo show. Usa /artista <

# Responder a cualquier mensaje con la IA
@bot.message_handler(func=lambda message: True)
def responder_mensaje(message):
    pregunta = message.text
    try:
        respuesta = genai.GenerativeModel('gemini-1.5-flash-latest').generate_content(pregunta)
        if respuesta and hasattr(respuesta, 'text'):
            bot.send_message(message.chat.id, respuesta.text)
        else:
            bot.send_message(message.chat.id, "No pude generar una respuesta. Intenta de nuevo más tarde.")
    except Exception as e:
        bot.send_message(message.chat.id, f"Error al generar la respuesta: {e}")

print("Bot de Telegram iniciado")
bot.polling()
```

6. **Manejo de errores**

   En todas las funciones principales un manejo de errores utilizando bloques try-except. Esto asegura que el bot no se caiga si ocurre un problema al hacer una solicitud a una API o al procesar un comando.

DIFICULTADES ENCONTRADAS

En el proceso de desarrollo del proyecto hemos encontrado diferentes dificultades. A continuación, un punteo de las mismas:

1. Lenguaje de las APIs

   Las APIs de Spotify y de Mercado Libre se encontraban en otros lenguajes de programación de los cuales no teníamos conocimientos. En este caso el desafío fue “traducir” dicho código a uno que sí conocemos: Python.

2. Acceso a ciertas APIs

   Durante el desarrollo del proyecto, uno de los principales desafíos fue la imposibilidad de acceder a ciertas APIs de pago que habrían mejorado significativamente la funcionalidad del bot. En particular, APIs como la de *Songkick*, que ofrece información sobre conciertos próximos de los artistas, requiere una suscripción paga para acceder a datos detallados y en tiempo real sobre los eventos. Dado que no contábamos con financiación para cubrir los costos de estas APIs, nos vimos limitados en la capacidad de ofrecer información completa sobre conciertos de los artistas, lo cual habría sido un valor añadido

para la plataforma. Si en el futuro el proyecto recibe financiación, la integración de Songkick podría enriquecer la experiencia del usuario al permitirle no solo descubrir artistas y productos de merchandising, sino también conocer los eventos y conciertos programados, brindando una experiencia más completa y personalizada.

3. Vencimiento del token de las APIs

   Relacionado con el punto anterior, nos hemos topado con la dificultad de, a la hora de acceder a la API, no lograrlo ya que el token tiene un tope de usos. Esto nos sucedió en particular con la API de Gemini.

4. Cambio en la documentación de la API de Spotify.

   En un principio, el código fue realizado con la documentación de la API de spotify cuyo nombre es “related artists” y cuya funcion principal era la de brindar artistas relacionados en base a otro. Inesperadamente la documentación de la API fue cambiada por su dueña, Spotify. Por ende su función principal ya no pudo ser ejecutada. Debido a esto, nos vimos en la obligación de cambiar el código y aplicar la API de Gemini para la mencionada función.

CONCLUSIONES

En aspectos generales, el proyecto cumple los objetivos planteados, pudiendo resolver los obstáculos presentados y concluir con su correcto funcionamiento.

Las tecnologías implementadas resultaron eficaces a la hora de utilizarlas y fueron de gran ayuda para construir el cuerpo del bot.

Este proyecto logró integrar eficazmente las APIs de Spotify, Mercado Libre y Gemini a través de un bot de Telegram, proporcionando a los usuarios una experiencia completa al ofrecer información sobre artistas y merchandising relacionado. Utilizamos la API de Spotify para obtener detalles sobre artistas y sus relacionados, y la API de Mercado Libre para mostrar productos de merchandising disponibles en la plataforma. Aunque la implementación de las APIs en Python supuso un desafío debido a la diferencia de lenguajes de programación, el uso de estas herramientas permitió construir una solución interactiva y accesible. A pesar de las dificultades encontradas durante el proceso de integración, el proyecto demuestra cómo se pueden combinar tecnologías para mejorar la retención del usuario, manteniéndolo más tiempo en la plataforma y alentándolo a

explorar productos relacionados.

PERSPECTIVAS

El bot logró cumplir las expectativas y superarlas. El camino hacia su desarrollo fue extenso, por momentos agobiante, pero al fin y al cabo gratificante. A su vez, consideramos que la idea del proyecto aporta el valor agregado de integrar en un solo lugar diferentes tecnologías que de no ser por el bot estarían separadas. Por esto mismo creemos que el código es funcional a lo que promete y ofrece una solución original a un problema particular.

El conocimiento tanto de las tecnologías como del lenguaje de programación utilizado fue fundamental a la hora de realizar el código que bajo nuestra perspectiva resultó completo y eficaz.

Por otro lado, la realización del proyecto nos brindó una mirada integral acerca del ejercicio de programar ya que hemos logrado aplicar nuestros conocimientos, incrementarlos y combinarlos entre sí para desarrollar una idea y un producto funcionales.

En síntesis, podemos establecer que las dificultades encontradas fueron superadas con éxito y aprendizaje, que el trabajo en equipo fue fundamental para el proceso y el desarrollo del chatbot, el cual resultó un producto y un servicio con altas oportunidades de ser financiado.