Presentation de l'outil Podcasts Tracker

Lyna BENYAHIA, Jason MOREL

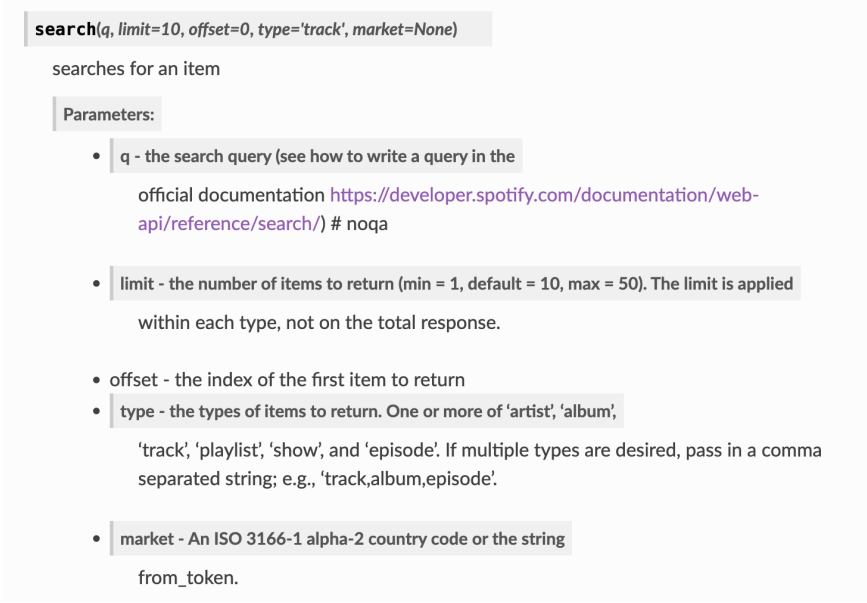
2023-03-07

Introduction

- Constat : il n'existe pas de moyens pour filtrer les résultats par durée sur Spotify
- Création d'un outil pour chercher des podcasts/émissions par durée
- Démonstration de l'outil

Mise en place du projet

1. Utilisation de Spotify pour notre projet à l'aide du package Python Spotipy
   Fonction principale que nous avons utilisé :



search(*q, limit=10, offset=0, type='track', market=None*)

searches for an item

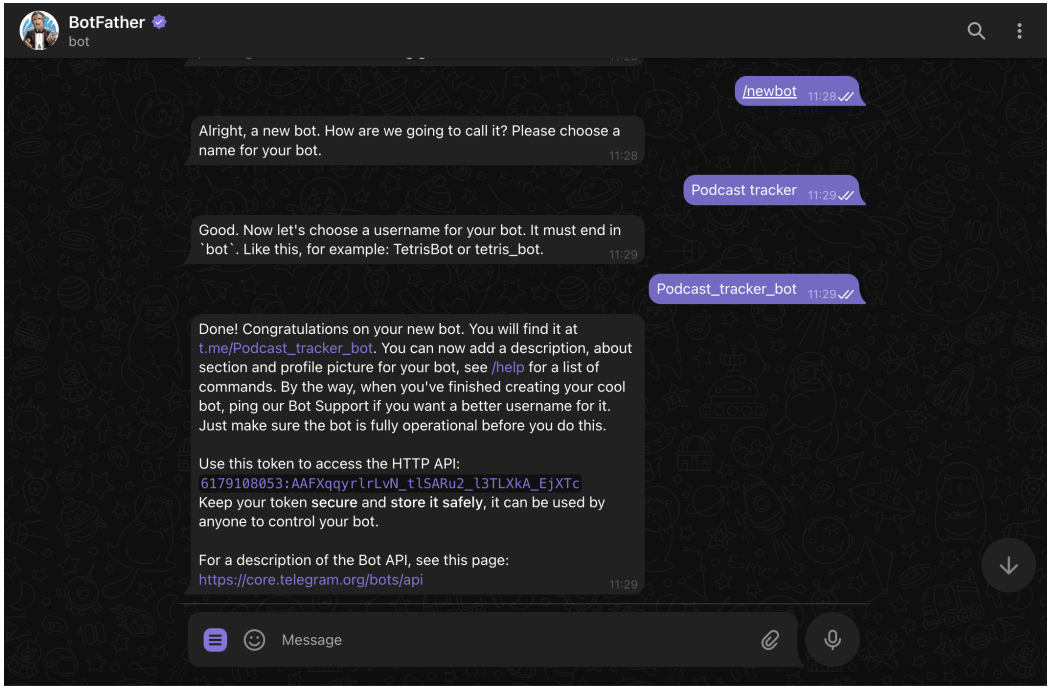
Parameters:

- q - the search query (see how to write a query in the official documentation https://developer.spotify.com/documentation/web-api/reference/search/) # noqa
- limit - the number of items to return (min = 1, default = 10, max = 50). The limit is applied within each type, not on the total response.
- offset - the index of the first item to return
- type - the types of items to return. One or more of 'artist', 'album', 'track', 'playlist', 'show', and 'episode'. If multiple types are desired, pass in a comma separated string; e.g., 'track,album,episode'.
- market - An ISO 3166-1 alpha-2 country code or the string from_token.

NB : distinction entre émission et épisode, impossibilité de faire des recherches par durée même depuis l'API

Mise en place du projet

2. Création d'un Bot Telegram pour envoyer les résultats afin d'obtenir les liens directement sur son téléphone



- Récupération du Token afin de connecter son code Python à son Bot Telegram

Premier problème rencontré

- Documentation très dense sur l'API Telegram, manque de temps pour créer un bot de qualité
  —> Envoi du résultat seulement sur Telegram

```python
# info sur telegram
TOKEN_telegram = "6179108053:AAFXqqyrlrLvN_tlSARu2_l3TLXkA_EjXTc" # obtenu en créant notre bot avec le telegram BotFather
#input arbitraire
chat_id = 0

# fonction pour envoyer des messages sur Telegram à un ID donné
def send_telegram_message(message, chat_id, TOKEN_telegram):
    url = f"https://api.telegram.org/bot{TOKEN_telegram}/sendMessage?chat_id={chat_id}&text={message}"
    response = requests.get(url)

    return response
```

  - Fonction pour envoyer un message sur Telegram (TelegramMessages sur Github)

Interface graphique

- Création d'une interface graphique à l'aide du package Tkinter (MessageVersion sur Github) :

MessageVersion.py

```
23
24
25
26
27  def submit_form():
28      # Récupération des valeurs sélectionnées par l'utilisateur
29      type_choice = type_var.get()
30      time_choice = time_var.get()
31      search_word = search_entry.get()
32      chat_id = chat_entry.get()
33
34      ############## PARTIE EPISODE ###################################
35      if type_choice == 1:
36          find_episode(search_word, time_choice, chat_id, TOKEN_telegram)
37
38      ############## PARTIE SHOW ######################################
39      if type_choice == 2:
40          find_shows(search_word, time_choice, chat_id, TOKEN_telegram)
41
42
43
44  # Création de la fenêtre et des widgets
45  root = tk.Tk()
46  root.title("Choix de podcasts") # CRÉATION DU TITRE DE LA FENÊTRE
47
48  chat_label = tk.Label(root, text="Démarrez une conversation avec 'userinfobot' sur Telegram.\nEntrez votre ID ici :").pack(pady=10) # DEMANDE DE L'ID DE L'UTILISATEUR
49  chat_entry = tk.Entry(root)
50  chat_entry.pack()
51
52  type_label = tk.Label(root, text="Que préférez-vous ?").pack(pady=10) # DEMANDE DE LA PRÉFÉRENCE ENTRE SHOW OU EPISODE
53  type_var = tk.IntVar()
54  type1_radio = tk.Radiobutton(root, text="Recevoir une liste d'épisodes uniques", variable=type_var, value=1).pack(anchor='w')
55  type2_radio = tk.Radiobutton(root, text="Recevoir une liste d'émissions (une émission contient plusieurs épisodes)", variable=type_var, value=2).pack(anchor='w')
56
57  time_label = tk.Label(root, text="Quel est le temps d'écoute que vous souhaitez ?").pack(pady=10) # DEMANDE DU TEMPS D'ÉCOUTE
58  time_var = tk.IntVar()
59  time1_radio = tk.Radiobutton(root, text="Moins de 5 minutes", variable=time_var, value=1).pack(anchor='w')
60  time2_radio = tk.Radiobutton(root, text="De 5 à 15 minutes", variable=time_var, value=2).pack(anchor='w')
61  time3_radio = tk.Radiobutton(root, text="De 15 à 30 minutes", variable=time_var, value=3).pack(anchor='w')
62  time4_radio = tk.Radiobutton(root, text="De 30 à 45 minutes", variable=time_var, value=4).pack(anchor='w')
63  time5_radio = tk.Radiobutton(root, text="Plus de 45 minutes", variable=time_var, value=5).pack(anchor='w')
64
65  search_label = tk.Label(root, text="Quel type de podcasts souhaitez-vous écouter ?\nEntrez le thème de votre choix :").pack(pady=10) # DEMANDE DU THÈME D'ÉCOUTE
66  search_entry = tk.Entry(root)
67  search_entry.pack()
68
69  submit_button = tk.Button(root, text="Valider", command=submit_form).pack(pady=10) # CRÉATION DU BOUTON "VALIDER"
70
71  root.mainloop() # MISE EN ROUTE DE L'INTERFACE
72
```

Traitement des émissions

- Show_treatment sur Github :

Traitement des épisodes

- Episode_treatment sur Github :

episode_treatment.py

```python
# Défini la durée minimale demandée par l'utilisateur
def min_for_episode(time_choice):
    if time_choice == 1:
        min_duration = 0
    elif time_choice == 2:
        min_duration = 300000
    elif time_choice == 3:
        min_duration = 900000
    elif time_choice == 4:
        min_duration = 1800000
    elif time_choice == 5:
        min_duration = 2700000

    return min_duration

# Défini la durée maximale demandée par l'utilisateur
def max_for_episode(time_choice):
    if time_choice == 1:
        max_duration = 300000
    elif time_choice == 2:
        max_duration = 900000
    elif time_choice == 3:
        max_duration = 1800000
    elif time_choice == 4:
        max_duration = 2700000
    elif time_choice == 5:
        max_duration = 10**1000

    return max_duration
# trouve une liste d'épisodes selon le thème inscrit, dont la durée est comprise entre la durée minimale et la durée maximale et en français
# envoie les noms et liens des épisodes à l'utilisateur sur Telegram

def find_episode(search_word, time_choice, chat_id, TOKEN_telegram):
    min_duration = min_for_episode(time_choice)
    max_duration = max_for_episode(time_choice)

    super_episode = sp.search(q=search_word, limit=50, type='episode', market='FR')
    selected_episodes = [episode for episode in super_episode['episodes']['items'] if min_duration <= episode['duration_ms'] <= max_duration and episode['language'] == 'fr']
    offset = 50
    while len(selected_episodes) < 4 and offset < super_episode['episodes']['total']:
        results = sp.search(q=f'{search_word}', limit=50, type='episode', market='FR', offset=offset)
        episodes = results['episodes']['items']
        selected_episodes += [episode for episode in episodes if min_duration <= episode['duration_ms'] <= max_duration and episode['language'] == 'fr']
        offset += 50

    if not selected_episodes:
        messagefinal = "Oups ! Aucun podcast ne correspond à vos critères. Veuillez entrer un autre thème."
    else:
        messagefinal = "Voici une liste de plusieurs podcasts correspondant à votre recherche :\n\n"
        for episode in selected_episodes[:3]:
            messagefinal += f"{episode['name']}\n{episode['external_urls']['spotify']}\n\n"

    send_telegram_message(messagefinal, chat_id, TOKEN_telegram)

    return messagefinal
```

Conclusion

- Difficultés qui émergent quand chacun travaille de son côté
- Extension possible : Bot Telegram complet