Leboncoin Scraper

Aimad Hamdaoui

22 Décembre 2024

Table des matières

1 Présentation générale

Le **Leboncoin Scraper** est un script en Python conçu pour automatiser la récupération d'annonces sur le site Leboncoin. Il utilise la bibliothèque `undetected-playwright` pour exécuter un navigateur Chromium, contourner d'éventuelles mesures anti-bots et extraire les informations d'annonces automobiles. Il envoie ensuite les nouvelles annonces correspondant à des mots-clés définis vers un **webhook Discord**.

2 Pré-requis

Avant de commencer, assurez-vous de disposer des éléments suivants :

— **Python 3.7+** installé sur votre machine ;
— Un **webhook Discord** valide (la configuration du webhook se fait dans les Paramètres de votre serveur Discord, dans la section *Intégrations* → *Webhooks*) ;
— La capacité d'installer et d'exécuter **Playwright** (en l'occurrence `undetected-playwright`) ;
— Un accès internet stable.

3 Installation

1. **Cloner ou télécharger** ce projet depuis le dépôt Git ou le dossier qui contient le script `.py`.
2. **Créer un environnement virtuel** (optionnel, mais recommandé) :

```
python -m venv venv
source venv/bin/activate   # Sur Linux/Mac
venv\Scripts\activate.bat  # Sur Windows
```

3. **Installer les dépendances** (notamment `requests`, `undetected-playwright`, etc.) :

```
pip install -r requirements.txt
```

4. **Installer les navigateurs** via Playwright :

```
playwright install
```

4 Configuration des fichiers

4.1 Fichier `keywords.txt`

Le fichier `keywords.txt` doit contenir la liste des **mots-clés** que vous souhaitez rechercher sur Leboncoin. Chaque mot-clé doit être sur une ligne distincte. Par exemple :

```
peugeot
renault
clio
golf
```

Le script va lire ce fichier au démarrage et ne prendra en compte que les annonces dont le titre contient au moins l'un de ces mots-clés (en minuscules ou majuscules, car la vérification est insensible à la casse).

4.2 Fichier `database.json`

Le fichier `database.json` sert de base de données locale pour stocker les liens des annonces déjà consultées ou déjà envoyées sur Discord. Ainsi, le script évite d'envoyer plusieurs fois la même annonce. S'il n'existe pas, il sera créé automatiquement lors de la première exécution du script.

5 Fonctionnement du script

Le script principal est un fichier `.py` qui utilise les bibliothèques `asyncio`, `undetected-playwright`, `requests` et `json`. Ci-dessous, un exemple de structure du code :

```python
import asyncio
import json
import requests
from pathlib import Path
from undetected_playwright.async_api import async_playwright

LEBONCOIN_SEARCH_URL = "https://www.leboncoin.fr/recherche?category=2&fuel=2&price=min-3500&mileage=min-250000&sort=time"
DISCORD_WEBHOOK_URL = "https://discord.com/api/webhooks/..."
DB_FILE = Path("database.json")
KEYWORDS_FILE = Path("keywords.txt")

def lire_keywords():
    """Lit les mots-cl s depuis le fichier keywords.txt."""
    if not KEYWORDS_FILE.exists():
        print(f"Fichier {KEYWORDS_FILE} introuvable. Veuillez le cr er et y ajouter des mots-cl s , un par ligne.")
        return []
    with KEYWORDS_FILE.open("r", encoding="utf-8") as f:
        return [line.strip().lower() for line in f if line.strip()]

def envoyer_vers_discord(titre, prix, annee, kilometrage, lien):
    """Envoie l'annonce sur Discord via un webhook."""
    try:
        embed = {
            "title": titre,
            "description": (
                f"**Prix** : {prix}\n"
                f"**Ann e** : {annee}\n"
                f"**Kilom trage** : {kilometrage}\n\n"
                f"[Voir l'annonce]({lien})"
            ),
            "color": 0x7289DA,
        }
        data = {
            "content": "Nouvelle annonce trouv e !",
            "embeds": [embed],
        }
        response = requests.post(DISCORD_WEBHOOK_URL, json=data)
        response.raise_for_status()
        print("Annonce envoy e  Discord avec succ s.")
    except requests.RequestException as e:
        print(f"Erreur lors de l'envoi  Discord : {e}")

async def main():
    """Relance la page de recherche toutes les 20 secondes apr s
```

```
        avoir fini de traiter les annonces."""
    keywords = lire_keywords()
    if not keywords:
        print("Aucun mot-cl  trouv . Le script s'arr te.")
        return

    # Lecture (ou cr ation) de la base de donn es locale
    if DB_FILE.exists():
        with DB_FILE.open("r", encoding="utf-8") as f:
            db = json.load(f)
    else:
        db = []

    async with async_playwright() as p:
        args = ["--disable-blink-features=AutomationControlled"]
        browser = await p.chromium.launch(headless=False, args=
           args)
        context = await browser.new_context(
            user_agent=(
                "Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
                "AppleWebKit/537.36 (KHTML, like Gecko) "
                "Chrome/91.0.4472.124 Safari/537.36"
            ),
            viewport={"width": 1280, "height": 720}
        )
        page = await context.new_page()

        try:
            while True:  # Boucle infinie
                try:
                    print("Navigation vers la page de recherche
                       ...")
                    await page.goto(LEBONCOIN_SEARCH_URL, timeout
                       =60000)
                    await page.wait_for_selector("a[data-test-id
                       ='ad']", timeout=60000)

                    # R cup ration de toutes les annonces
                    annonces = await page.query_selector_all("a[
                       data-test-id='ad']")
                    print(f"Nombre total d'annonces trouv es : {
                       len(annonces)}")

                    for annonce in annonces:
                        try:
                            # Titre
                            titre_elem = await annonce.
                               query_selector("p[data-qa-id='
                               aditem_title']")
                            if not titre_elem:
                                continue
```

```
87                titre = await titre_elem.inner_text()
88
89                # V rifie si le titre contient l'un
                      des mots-cl s
90                if not any(keyword in titre.lower()
                      for keyword in keywords):
91                    continue
92
93                # Lien
94                lien = await annonce.get_attribute("
                      href")
95                if not lien:
96                    continue
97                lien_complet = f"https://www.
                      leboncoin.fr{lien}"
98
99                # Prix
100               prix_elem = await annonce.
                      query_selector("p[data-test-id='
                      price']")
101               prix = await prix_elem.inner_text()
                      if prix_elem else "Non sp cifi  "
102
103               # Ann e et kilom trage
104               divs = await annonce.
                      query_selector_all("div.relative.h-
                      full.whitespace-nowrap")
105               if len(divs) >= 2:
106                   # Ann e
107                   annee_elem = await divs[0].
                          query_selector("p:last-child")
108                   annee = await annee_elem.
                          inner_text() if annee_elem else
                           "Non sp cifi e"
109
110                   # Kilom trage
111                   km_elem = await divs[1].
                          query_selector("p:last-child")
112                   kilometrage = await km_elem.
                          inner_text() if km_elem else "
                          Non sp cifi  "
113               else:
114                   annee = "Non sp cifi e"
115                   kilometrage = "Non sp cifi  "
116
117               # V rifie si d j  pr sent dans la
                      DB
118               if lien_complet in db:
119                   continue
120
121               # Ajout dans la DB et envoi Discord
```

```
122                                 db.append(lien_complet)
123                                 envoyer_vers_discord(titre, prix,
                                        annee, kilometrage, lien_complet)
124
125                                 print(f"Nouvelle annonce ajout e : {
                                        titre} | {prix} | {annee} | {
                                        kilometrage}")
126                             except Exception as e:
127                                 print(f"Erreur de traitement d'une
                                        annonce : {e}")
128
129                         # Sauvegarde de la DB
130                         with DB_FILE.open("w", encoding="utf-8") as f
                                :
131                             json.dump(db, f, ensure_ascii=False,
                                    indent=4)
132
133                         print("Termin  pour cette boucle. Relance de
                                la recherche dans 20 secondes...")
134                         await asyncio.sleep(20)
135
136                 except Exception as e:
137                     print(f"Erreur lors de la navigation ou du
                            parsing : {e}")
138                     await asyncio.sleep(20)
139
140         finally:
141             await browser.close()
142
143 if __name__ == "__main__":
144     asyncio.run(main())
```

Voici les différentes étapes clés :

5.1 Lecture des mots-clés

La fonction lirekeywords() ouvre le fichier keywords.txt et en charge les mots-clés, un par ligne. Chaque mot-clé est converti en minuscules pour simplifier la vérification.

5.2 Chargement de la base de données locale

Le script vérifie l'existence d'un fichier `database.json`. S'il existe, il est chargé en mémoire pour récupérer la liste des liens déjà envoyés. S'il n'existe pas, une liste vide est initialisée.

5.3 Lancement du navigateur avec `undetected-playwright`

Le script lance une instance de Chromium en mode `headless=False` (navigateur visible). Des arguments sont passés pour contourner les techniques de détection de bot. Une page est ensuite ouverte dans ce contexte.

5.4 Collecte et traitement des annonces

Le script navigue vers l'URL de recherche Leboncoin (stocker dans `LEBONCOIN_SEARCH_URL`). Une fois la page chargée, il recherche toutes les balises correspondantes aux annonces (`a[data-test-id='ad']`). Pour chacune d'entre elles :

— Il récupère le titre, le lien, le prix, l'année et le kilométrage ;
— Il vérifie si le titre contient l'un des mots-clés.

5.5 Filtrage des annonces par mots-clés

Le script ne conserve que les annonces dont le titre contient au moins un mot-clé du fichier `keywords.txt`. La correspondance se fait en minuscules.

5.6 Envoi vers Discord

Si l'annonce n'est pas déjà dans la base de données (`database.json`), la fonction `envoyer_vers_discord()` est appelée. Elle construit un objet `embed` (spécifique à Discord) et poste une requête HTTP vers le webhook défini dans `DISCORD_WEBHOOK_URL`.

5.7 Mise à jour de la base de données

Le lien de l'annonce est ensuite ajouté à la liste en mémoire, puis la base de données est mise à jour en écrivant dans `database.json`. Cela empêche toute future duplication d'annonces.

6 Exécution du script

Pour exécuter le script, il suffit de lancer :

```
1 python votre_script.py
```

Assurez-vous d'avoir bien configuré vos fichiers `keywords.txt` et `database.json` (ou de laisser le script créer le second) ainsi que votre `DISCORD_WEBHOOK_URL` dans le code.

Le script reste en boucle **infinie** ; toutes les 20 secondes, il recharge la page, inspecte les annonces, envoie les nouvelles annonces correspondantes, puis attend de nouveau 20 secondes.

7 Conseils et conclusions

— Le script étant dans une boucle infinie, vous pouvez l'arrêter manuellement avec `CTRL + C`.
— Vous pouvez adapter l'URL de recherche `LEBONCOIN_SEARCH_URL` selon vos propres critères (catégorie, filtre de prix, kilométrage, etc.).
— Ajustez la `DISCORD_WEBHOOK_URL` avec celle de votre propre serveur Discord.
— N'hésitez pas à personnaliser la fréquence de mise à jour (actuellement 20 secondes via `asyncio.sleep(20)`).
— Si vous exécutez le navigateur en `headless=True`, vous ne verrez pas l'interface, mais cela fonctionnera en tâche de fond.