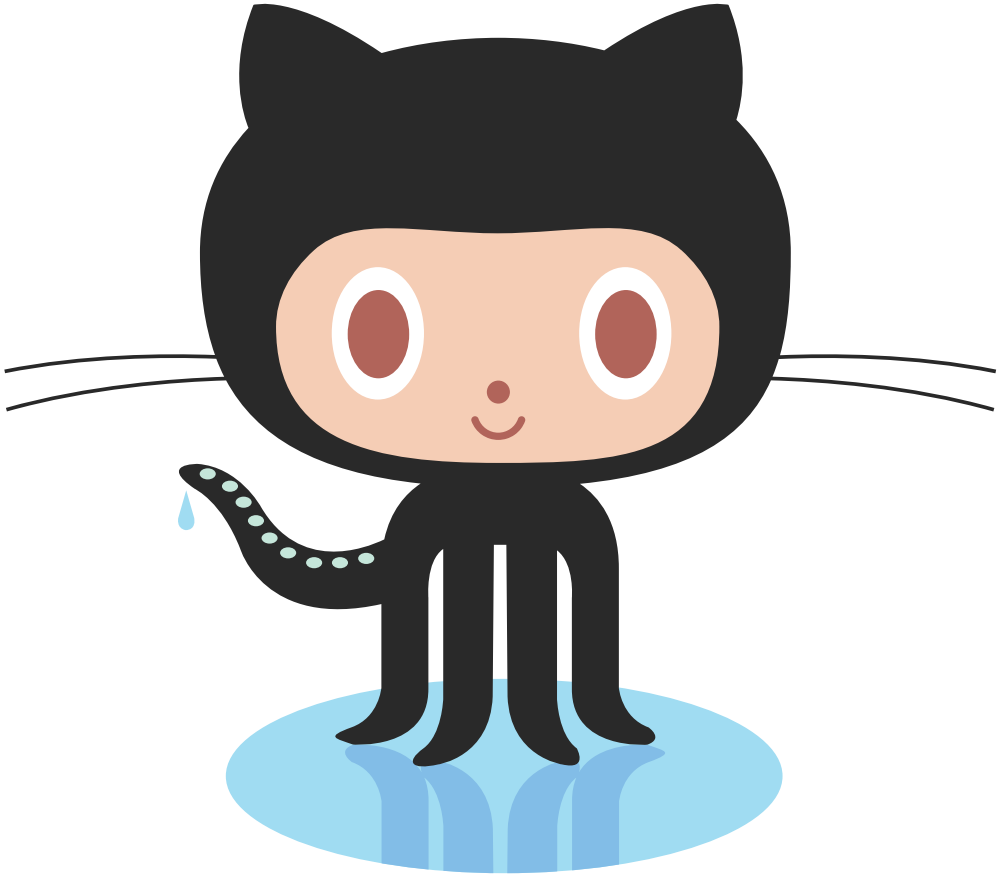
Scraping GitHub Project



Created by : AMIN ELFAQUIRI - 16/06/2023

Cliquez ici pour accéder au projet sur GitHub.

Introduction :

Ce document présente les étapes réalisées et les difficultés rencontrées lors de la réalisation du projet de collecte de données depuis GitHub. L'objectif était de collecter des informations pertinentes sur les dépôts de code hébergés sur GitHub pour mieux comprendre les tendances de développement, les langages de programmation les plus utilisés et identifier des projets intéressants.

web Scraping

Définition :

Le web scraping est une technique utilisée pour extraire des données à partir d'un site Web. Il peut être utilisé pour collecter des données en grande quantité et de manière automatisée, afin de gagner du temps et privilégier l'analyse des données collectées et l'exploitation des résultats obtenus.

Les méthodes :

Utilisation de l'API :

Avantages :

- Accès direct aux fonctionnalités .
- Données actualisées en temps réel.
- Flexibilité dans la récupération des informations.

Limitations :

- Limites de fréquence possibles.
- Authentification requise pour certaines fonctionnalités.

Utilisation de bibliothèques Python comme PyGitHub :

Avantages :

- Interface conviviale pour interagir avec l'API GitHub.
- Fonctionnalités supplémentaires pour simplifier les tâches courantes.

Limitations :

- Dépendance externe à installer et à importer.
- Mises à jour régulières nécessaires.

Étapes réalisées :

1. Définition des objectifs du projet.
2. Recherche et documentation .
3. lire la politique du web scraping de github.
4. essayer de récupérer des dépôts en utilisant la bibliothèque PyGithub .
5. essayer de récupérer des dépôts en utilisant la bibliothèque requests .

6. Exporter les données au format csv.

En utilisant la bibliothèque PyGithub :

Les étapes présentes dans le code :

1 - L'importation des bibliothèques nécessaires :

PyGithub : qui permet d'interagir avec l'API (interface de programmation d'application) de GitHub.

datetime : qui fournit des classes pour manipuler les dates et les heures en Python.

Pandas : Je vais utiliser la bibliothèque pandas pour créer un dataframe et l'enregistrer sous forme de fichier CSV.

2- Obtenir un jeton via l'API de Github .

3- Je crée une liste pour stocker les informations que j'obtiens

4- Je crée des variables pour limiter le nombre de dépôts obtenus chaque jour.

5 - Je crée une boucle de dates qui commence le 1er janvier 2023 et se termine aujourd'hui .

6 - J'utilise la bibliothèque PyGithub pour m'authentifier auprès de l'API GitHub en utilisant un jeton d'accès (access token). Ensuite, j'obtiens les 1000 meilleurs dépôts (repos) ayant le plus de stars créés ce jour-là.

7 - Je crée une boucle et à l'intérieur de celle-ci, je crée un dictionnaire contenant toutes les informations nécessaires pour le dépôt ciblé. Ensuite, j'ajoute ce dictionnaire à une liste préalablement créée.

Exécution du code :

```
2023-01-01
############################## month => 2023-06 ##############################
1 facebookresearch/audiocraft
2 baichuan-inc/baichuan-7B
3 Licoy/ChatGPT-Midjourney
4 SysCV/sam-hq
5 facebookresearch/ijepa
6 Mintplex-Labs/anything-llm
7 fractureiser-investigation/fractureiser
8 KenneyNL/Adobe-Alternatives
9 qianqianwang68/omnimotion
10 InternLM/InternLM-techreport
```

```
11 pennyliang/cku
12 Not-Quite-RARBG/main
13 MCRcortex/nekodetector
14 kargisimos/offensive-bookmarks
15 Sh4yy/cloudflare-email
16 ZeroMemoryEx/Terminator
17 axodox/axodox-machinelearning
18 pen4uin/java-memshell-generator
19 antfu/raycast-multi-translate
20 krzysztofzablocki/Swift-Macros
21 n4ze3m/dialoqbase
22 Vahe1994/SpQR
23 mit-han-lab/llm-awq
...
Error message: 403 {"message": "API rate limit exceeded for user ID 81482544.",
 "documentation_url": "https://docs.github.com/rest/overview/resources-in-the-rest-api#rate-limiting"}
638 joshuatazrein/obsidian-time-ruler
Error retrieving repository: joshuatazrein/obsidian-time-ruler
Error message: 403 {"message": "API rate limit exceeded for user ID 81482544.",
 "documentation_url": "https://docs.github.com/rest/overview/resources-in-the-rest-api#rate-limiting"}
```

" Je récupère Number large 600 ou 1020 dépôts . Ensuite, j'ai dépassé la limite de taux de l'API "

Utilisez la bibliothèque Requests :

Je crée un code qui effectue une recherche sur l'API GitHub pour obtenir les dépôts les plus populaires, classés par nombre d'étoiles, créés chaque jour à partir de la date de début jusqu'à la date actuelle. Les informations des dépôts sont extraites et stockées dans un dictionnaire, puis ajoutées à la liste all_repo. Pour respecter les limites de taux de l'API, j'ajoute une pause de 1 seconde entre chaque requête afin d'éviter de dépasser la limite.

Les étapes présentes dans le code :

1 - L'importation des bibliothèques nécessaires : timet , requests

2- Création d'une liste pour stocker les dépôts

3 - Définition des variables de comptage et de limite

4 - Définition des dates de début et de fin de recherche

5 - Boucle pour parcourir chaque jour entre la date de début et la date actuelle

6 - Recherche des dépôts sur GitHub

7 - Boucle pour parcourir chaque dépôt trouvé

8 - Ajout des informations du dépôt à la liste

9 - Vérification de la limite de dépôts atteinte

10 - Gestion des erreurs .

11 - crée un objet DataFrame à partir de la liste .

	Author	name	full_name	languages	stars	forks	watchers	issues	created_date	last_update	
0	facebookresearch	audiocraft	facebookresearch/audiocraft	Python	6589	573	6589	73	2023-06-08T06:41:36Z	2023-06-15T18:03:41Z	https://github.com/facet
1	Licoy	ChatGPT-Midjourney	Licoy/ChatGPT-Midjourney	TypeScript	2564	728	2564	10	2023-06-05T10:35:13Z	2023-06-15T17:39:54Z	https://github.com/Lic
2	baichuan-inc	baichuan-7B	baichuan-inc/baichuan-7B	Python	1875	111	1875	17	2023-06-14T10:57:51Z	2023-06-15T18:04:38Z	https://github.com/ba
3	SysCV	sam-hq	SysCV/sam-hq	Python	1640	49	1640	11	2023-06-01T02:04:54Z	2023-06-15T17:08:55Z	https://gith
4	Mintplex-Labs	anything-llm	Mintplex-Labs/anything-llm	JavaScript	1038	123	1038	13	2023-06-04T02:29:14Z	2023-06-15T17:48:10Z	https://github.com/Mir
...	...	...	...	...	...	...	...	...	...	...	
165995	BinaryBirds	swift-macros	BinaryBirds/swift-macros	Swift	11	0	11	0	2023-06-08T06:02:09Z	2023-06-12T18:27:57Z	https://github.com/B
165996	FerlyXyn	Fb-Toolkit	FerlyXyn/Fb-Toolkit	Python	11	2	11	0	2023-06-09T13:09:47Z	2023-06-09T14:23:56Z	https://github.c
165997	Ryota-Kawamura	Building-Systems-with-the-ChatGPT-API	Ryota-Kawamura/Building-Systems-with-the-ChatG...	Jupyter Notebook	11	7	11	0	2023-06-03T15:18:36Z	2023-06-15T04:25:36Z	https://github.com/Ryo
165998	ZephrFish	MoveIT-WebShellCheck	ZephrFish/MoveIT-WebShellCheck	Python	11	1	11	0	2023-06-01T23:30:36Z	2023-06-13T12:06:05Z	https://github.c
165999	zurmokeeper	excelize	zurmokeeper/excelize	JavaScript	11	1	11	0	2023-06-04T06:30:03Z	2023-06-14T05:15:40Z	https://github.com

166000 rows × 11 columns

On a extrait 166 000 lignes en 86 minutes.

12 - Exporter ce DataFrame sous la forme d'un fichier CSV.

Difficultés rencontrées lors de la collecte de données :

Limites de l'API :

En utilisant l'API GitHub, j'ai constaté qu'il y avait une limite de 1000 dépôts par requête. Cela signifie que je pouvais seulement récupérer les 1000 premiers dépôts correspondant à ma requête.

Temps d'exécution :

Étant donné que je collectais des données sur une période de temps étendue, le temps d'exécution était un défi.

Conclusion :

En conclusion, ce projet de collecte de données depuis GitHub a présenté plusieurs défis, tels que la limite de l'API à 1000 dépôts par requête, la gestion des erreurs, le temps d'exécution et la manipulation des données. Malgré ces difficultés, j'ai pu acquérir une expérience précieuse en gestion de l'API, en optimisation du code et en manipulation des données. Avec patience et ajustements, j'ai réussi à mener à bien ce projet.