Scraping GitHub

SENHAJI Hamza

Rédigé le 16 juin 2023 à 16:16

1 Description de Brief :

Notre entreprise est intéressée par l'obtention d'informations précieuses sur les dépôts de code hébergés sur GitHub.

L'objectif est de mieux comprendre les tendances de développement, les langages de programmation les plus utilisés, d'identifier des projets intéressants, de suivre les évolutions technologiques et d'explorer de potentielles opportunités de collaboration.

Pour atteindre ces objectifs, nous souhaitons développer un projet data qui nous permettra d'identifier les informations pertinentes sur GitHub.

En tant que développeur de données, votre première tâche consistera à naviguer sur le web afin d'effectuer une collecte de données GitHub pertinente.

2 Description de la source data : "Github"

GitHub est une plateforme web populaire destinée à l'hébergement de dépôts de contrôle de version et à la collaboration entre développeurs.

Que vous soyez développeur individuel, membre d'une équipe ou contributeur open source, GitHub offre une plateforme puissante et collaborative pour gérer et partager vos projets de développement.

Voici une courte présentation de GitHub :

- **Contrôle de version** : GitHub utilise Git, un système de contrôle de version distribué, pour suivre les modifications apportées au code et aux fichiers. Git permet aux développeurs de gérer différentes versions de leurs projets, de travailler sur des branches distinctes et de fusionner les modifications.
- **Hébergement de code** : GitHub offre un service d'hébergement de dépôts distant, ce qui signifie que vous pouvez stocker vos projets de développement sur des serveurs GitHub accessibles en ligne. Cela facilite le partage et la collaboration avec d'autres développeurs.
- **Collaboration** : GitHub facilite la collaboration entre les membres d'une équipe de développement. Les fonctionnalités telles que les demandes de tirage (pull requests), les problèmes (issues) et les commentaires permettent aux développeurs de discuter des modifications, de suggérer des améliorations et de résoudre les problèmes ensemble.
- **Intégration continue** : GitHub propose des intégrations avec des outils d'intégration continue tels que Jenkins, Travis CI et CircleCI. Cela permet d'automatiser les tests, les builds et le déploiement de votre code à chaque modification.
- **Communauté open source** : GitHub est une plateforme très utilisée par la communauté open source. Elle offre des fonctionnalités spécifiques pour les projets open source, telles que la gestion des contributions, les licences et les flux de travail spécifiques aux projets.

— **Gestion de projets** : GitHub propose des outils de gestion de projets pour organiser les tâches, suivre les problèmes et planifier les étapes de développement. Vous pouvez utiliser des tableaux Kanban, des projets GitHub ou intégrer des outils de gestion de projets tiers.

3 Bibiothèques utilisées :

3.1 Requests

Requests est une bibliothèque Python populaire qui simplifie l'envoi de requêtes HTTP. Elle offre une interface simple et intuitive pour effectuer des requêtes GET, POST, PUT, DELETE et d'autres méthodes courantes, ainsi que pour gérer les paramètres, les en-têtes et les cookies.

3.2 BeautifulSoup

Beautiful Soup est une bibliothèque populaire en Python utilisée pour le web scraping et l'analyse de documents HTML ou XML. Elle fournit une interface simple et intuitive pour extraire des informations à partir de pages web.

3.3 pyGithub

PyGitHub est une bibliothèque Python qui fournit une interface de haut niveau pour interagir avec l'API GitHub. Elle simplifie l'accès et la manipulation des dépôts GitHub, des problèmes (issues), des demandes de tirage (pull requests) et d'autres ressources de GitHub de manière programmable.

4 Contraintes :

4.1 Contraintes

— **Limite de fréquence des requêtes** : GitHub limite le nombre de requêtes que vous pouvez effectuer sur son API dans une période donnée. En général, les limites sont fixées à un certain nombre de requêtes (actuellement 1000).

— **Authentification** : Pour accéder à certaines fonctionnalités de l'API GitHub, vous devrez peut-être vous authentifier. L'utilisation d'un jeton d'accès personnel (personal access token) est souvent recommandée pour l'authentification.

— **Limite de pagination** : Lorsque vous récupérez des listes d'éléments (par exemple, les dépôts, les problèmes, etc.), l'API GitHub utilise la pagination pour limiter le nombre d'éléments renvoyés par requête. Vous devrez gérer la pagination en utilisant les paramètres de pagination fournis par l'API pour récupérer tous les éléments souhaités.

4.2 Solutions :

Pour surmonter les contraintes lors de l'extraction d'informations de GitHub, voici quelques suggestions :

— Respectez les limites de fréquence : Assurez-vous de connaître les limites actuelles de l'API GitHub et respectez-les

— Utilisez l'authentification

— Gérez la pagination : Lorsque vous récupérez des listes d'éléments paginées, assurez-vous d'implémenter la gestion de la pagination pour récupérer tous les éléments.

— Utilisez la mise en cache : Mettez en place une stratégie de mise en cache pour éviter de faire des requêtes redondantes à l'API GitHub.

5 Pipeline :

5.1 Objectif :

Notre objectif et de collecter des informations sur des "repositories" de github pour avoir des informations sur les languages les plus utilisées dans differentes nouvelles technologies.
On veut collecter 10000 dépots et ses information comme première application.

5.2 Méthode et résultats :

1. Nous commençons par chercher sur github par le mot clès "repositories", On obtient 100 pages, chacune contient 10 dépot.

2. On utilise la bibliothèque "requests" et "Beautifulsoup" pour extraire le maximun des resultats à partir des pages en loopant par language (respectivement par page).

```
url_languages = ['HTML', 'Ruby', 'Python', 'shell', 'JavaScript', 'Java', 'Jupyter notebook', 'C++', 'R', 'C#' ]
```

```
#create a dictionnary to stock the data
repos = { 'label': [],
          'link' : []
         }
#Loop by page and languages
for languages in url_languages:
    x = languages
    for i in range(100):
        url = f"https://github.com/search?l={x}&p={i}&q=repositories&type=Repositories"
        #
        response = requests.get(url)
        if response.status_code == 200:
            # Proceed with extracting information from the response
            html_content = response.text
            # Create a BeautifulSoup object to parse the HTML
            soup = BeautifulSoup(html_content, 'html.parser')
            # Extract the desired information from the HTML using BeautifulSoup methods
            for repo in soup.select('.repo-list-item .v-align-middle'):
                name = repo.text.strip()
                repos['label'].append(name)
                link = f"https://github.com/{name}"
                repos['link'].append(link)
                time.sleep(15)
```

On obtient un data de deux colonnes (user, link), contient 18000 ligne.

0	user	link
0	Python-GUI-Project	https://github.com/Aashishkumar123/Python-GUI-...
1	Hands-on-Microservices-with-Python-Order-Service	https://github.com/PacktPublishing/Hands-on-Mi...
2	Amazon-Previous-OA-questions	https://github.com/NoobSolver/Amazon-Previous-...
3	Microsoft.Xrm.Data.PowerShell.Samples	https://github.com/seanmcne/Microsoft.Xrm.Data...
4	chocofactory	https://github.com/chocomintapp/chocofactory
5	Final_Projects	https://github.com/epam-js-march-2019/Final_Pr...
6	ws-0100-codepen-copy	https://github.com/version1-workspace/ws-0100-...
7	NowIamFeelingThat	https://github.com/VoQn/NowIamFeelingThat
8	Defeat-JAVA	https://github.com/sam-tripathi/Defeat-JAVA
9	code_repo	https://github.com/falwat/code_repo

3. Aprés l'extraction des noms et des liens des dépots, On utilise pyGithub pour extraire les informations souhaitées autour des dépots.

```
# Public repository
g = Github()
g = Github("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")

dict_repo = {"Full name" : [ ],
                "Description" : [ ],
                "Date created" : [ ],
                "Date of last push" : [ ],
                "Home Page" : [ ],
                "Language:" : [ ],
                "Number of forks" : [ ],
                "Number of stars" : [ ]
            }
```

```
count = 0
for link in data_repo["link"]:
    if (count < 50):
        user = link.replace("https://github.com/", "")
        repo = g.get_repo(user)

        # repository full name
        #print("Full name:", g.get_repo(repo).full_name)
        dict_repo["Full name"].append(repo.full_name)

        # repository description
        #print("Description:", g.get_repo(repo).description)
        dict_repo["Description"].append(repo.description)

        # the date of when the repo was created
        #print("Date created:", g.get_repo(repo).created_at)
        dict_repo["Date created"].append(repo.created_at)

        # the date of the last git push
        #print("Date of last push:", g.get_repo(repo).pushed_at)
        dict_repo["Date of last push"].append(repo.pushed_at)

        # home website (if available)
        #print("Home Page:", g.get_repo(repo).homepage)
        dict_repo["Home Page"].append(repo.homepage)

        # programming language
        #print("Language:", g.get_repo(repo).language)
        dict_repo["Language:"].append(repo.language)

        # number of forks
        #print("Number of forks:", g.get_repo(repo).forks)
        dict_repo["Number of forks"].append(repo.forks)

        # number of stars
        #print("Number of stars:", g.get_repo(repo).stargazers_count)
        #print("-"*50)
        dict_repo["Number of stars"].append(repo.stargazers_count)
        count = count + 1
        time.sleep(30)
    else:
        count = 0
```

4. Finalement on obtient une data qui contient 9 colonnes :
   - Full name
   - Description
   - Date created
   - Date of last push
   - Home Page
   - Language
   - Number of forks
   - Number of stars

Full name	Description	Date created	Date of last push	Home Page	Language:	Number of forks	Number of stars
Aashishkumar123/Python-GUI-Project	A Repositry that contains 20+ GUI Projects on ...	2020-06-23 10:21:50	2023-06-11 14:30:45	https://github.com/Aashishkumar123/Python-GUI-...	Python	191	387
PacktPublishing/Hands-on-Microservices-with-Py...	Repositry for Order Services	2018-11-02 11:05:08	2023-05-22 21:36:48	NaN	Python	23	14
NoobSolver/Amazon-Previous-OA-questions	This repositry contains all the Previous year ...	2021-01-03 14:20:49	2021-01-03 14:25:18	NaN	C++	50	120
seanmcne/Microsoft.Xrm.Data.PowerShell.Samples	This is samples repositry for Microsoft.Xrm.Da...	2015-11-09 15:56:19	2022-06-23 18:13:47	https://github.com/seanmcne/Microsoft.Xrm.Data...	PowerShell	22	48
chocomintapp/chocofactory	This is chocofork repositry	2021-03-13 04:43:58	2022-10-27 05:24:09	NaN	TypeScript	4	19
...	...	...	...	...	...	...	...
psujit775/terraform-templates	This Repositry contains list of terraform temp...	2022-03-19 18:20:13	2023-02-16 14:06:54	NaN	HCL	2	0
Sahil624/Python-Basics	This is Repositry of Python Basics.	2017-02-16 05:05:15	2020-10-03 10:02:52	NaN	Python	2	0
ManjunathACG/Hive-Repositries	NaN	2018-09-27 10:26:29	2018-09-27 10:26:30	NaN	NaN	0	0
YoungForm/GIT_repositries	NaN	2019-02-18 02:48:32	2019-02-18 05:27:18	NaN	NaN	0	0

FIGURE 1 – data finale

6 Conclusion et perspectives

Dans ce projet, on a essayé de collecter la data sur les dépots de github en utilisant différentes bibliothèques de web scraping (requests, pyGithub, Beautifulsoup). dont laquelle on va l'utiliser après pour faire la prédiction des languages les plus utilisées dans differentes technologies.