Projet InfoCommerce

Plan :

French

Projet InfoCommerce

Fichiers du code disponibles sur mon profil GitHub : **ValentinVigouroux**, repository : « InfoCommerce »

Introduction

Dans ce projet, j'ai extrait des données d'un site public répertoriant des indicateurs financiers des entreprises françaises à l'aide de l'API du site internet **annuaire-entreprise.fr**.

Motivation du projet

Le but du projet est de déterminer quels sont les secteurs et entreprises les plus dynamiques économiquement à partir de données réelles agrégées pour la France, et de dégager des points communs à ces entreprises.
En parallèle, l'objectif était de développer mes compétences en **analyse de données , web scraping et analyse financière.**

Justification du Scraping

Une API officielle est disponible sur **annuaire-entreprise.fr**, cependant après demande de la clé API, aucune réponse ne m'a été accordée. J'ai donc utilisé ce modèle de scraping.
Je me suis rendu compte que les données financières étaient disponibles dans la requête d'URL d'API effectuée par la page et visible dans l'onglet **Network** de l'inspecteur du site.

Description du code

Il s'agit d'un spider **Scrapy (s.py)**. À partir d'une base de données SQL contenant une liste d'entreprises que j'ai construite, le scraper extrait en particulier le **SIREN** de chaque entreprise. Il récupère le contenu de la requête d'URL d'API (« https://data.economie.gouv.fr/api/records/1.0/search/?dataset=ratios_inpi_bce&q=siren%3A » {siren} en format JSON) via la méthode GET pour chaque entreprise (construction de l'URL contenant le SIREN).

Les informations sont classées, subissent des pré-traitements, puis sont ré-exportées dans un fichier CSV. J'utilise ensuite ces données dans un **dashboard Power BI**.

Paramètres dans settings.py :

- La vitesse est ajustée à un rythme relativement lent pour ne pas surcharger annuaire-entreprise.fr.
- Le paramètre **Autothrottle** : adapte la vitesse du scraper au trafic du site pour éviter de le surcharger et adopter un comportement plus naturel (moins détectable).
- Des **user-agents** pris aléatoirement dans une liste sont associés à chaque requête pour rendre la détection du robot plus difficile.
- Le fichier robots.txt est ignoré et les cookies sont désactivés → scraping qui viole les termes du fichier robots.txt de annuaire-entreprise.fr.

Mise en garde sur le Scraping

Scraper autrement que par l'API officielle est interdit par les règles de « annuaire-entreprise.fr. »
Ce projet a une **vocation d'apprentissage et de curiosité intellectuelle**.
Je vous invite à ne pas utiliser ce code de façon abusive.
Je distribue uniquement ce code pour présenter mes compétences sur le marché du travail.
Mon code est l'œuvre d'un novice et est intensif en requêtes ; après environ dix mille entreprises scrappées le site bloquera votre accès pour une journée. En cas d'abus, vous vous exposez à un bannissement du site voire à des poursuites en raison de l'impact d'une surcharge de requêtes.

Attention : le code extrait les informations sur une liste de 2,1M d'entreprises : **Presser "Ctrl+c" deux fois dans le terminal pour arrêter le code manuellement.**

Étapes pour exécuter le code (pour les profanes)

J'utilise **Anaconda Navigator**, qui facilite l'installation et la gestion des librairies.

1. Ouvrir Anaconda (gestionnaire d'environnement).
2. Sélectionner un environnement contenant les librairies **Scrapy** et **Protego** (en plus de celles par défaut).
3. Télécharger les fichiers (spider + fichier Excel liste d'entreprises contenant les SIREN). (**repository infoCommerce**)
4. Lancer votre éditeur de code (**VS Code**) depuis votre gestionnaire d'environnement.
5. Ouvrir les fichiers dans votre éditeur de code.

6. Ouvrir le **spider s.py**.
7. Dans le terminal, se placer dans le dossier spiders (« cd api/spiders »)
8. Lancer le programme avec la commande :

   scrapy runspider s.py -o resultats.csv -s FEED_EXPORT_ENCODING=utf-8-sig -s FEED_EXPORT_FIELDS_DELIMITER=";" -s JOBDIR=crawls/save1

Cette commande exportera les données dans un fichier Excel tout en gardant en mémoire les données déjà extraites si vous devez le faire en plusieurs fois.

Attention : le code extrait les informations sur une liste de 2,1M d'entreprises : **Presser "Ctrl+c" deux fois dans le terminal pour arrêter le code manuellement.**

Informations sur la nature des données

Dans ce projet, je me suis concentré sur les **entreprises purement commerciales**, c'est-à-dire celles qui vendent et produisent des biens et services.
Je pars d'une liste des entreprises en France disponible sur **l'INSEE (« StockUniteLegale_4colonnes.csv »)** Elle contient 4 Millions d'entreprises.
Je filtre :

- les entreprises commerciales (catégorie juridique « 5_ »…)
- certains secteurs (sociétés purement financières, assurance, immobilières…) pour lesquels les indicateurs financiers recueillis ne sont pas pertinents pour analyser l'activité.

(ce code est disponible dans le fichier : « codeSQLgrosfichier.sql »)

Après filtrage, la liste contient environ **2,1 millions d'entreprises**.
Le code puise directement dans cette liste de SIREN et extrait les informations financières suivantes pour chaque entreprise :

Exemple :

{

"taux_d_endettement": 0.0,

"credit_fournisseurs_jours": 100.118,

"marge_ebe": 32.125,

"caf_sur_ca": 15.535,

"poids_bfr_exploitation_sur_ca_jours": 129.224,

"marge_brute": 7413504,
"resultat_courant_avant_impots_sur_ca": 18.651,
"date_exercice": "2021-12-31",
"ebit": 1789388,
"rotation_des_stocks_jours": 76.579,
"siren": "310865597",
"chiffre_d_affaires": 9630088,
"confidentiality": "Public",
"resultat_net": 1374249,
"ebe": 3095627,
"type_bilan": "C",
"poids_bfr_exploitation_sur_ca": 35.895,
"ratio_de_liquidite": 144.734,
"credit_clients_jours": 60.701
}

En plus d'informations de secteurs et d'effectifs qui proviennent du fichier source.

Points d'amélioration du code

De nombreux axes d'amélioration sont possibles, en particulier :

- Rendre le code moins intensif en requêtes pour scraper plus rapidement et sans surcharger le site.
- Arrêter le code automatiquement en cas de blocage du site (réponse 429).
- Comprendre comment le spider choisit l'ordre des SIREN et adapter cette méthode pour assurer un **tableau statistiquement représentatif** de l'activité française.
- …

Contact

Pour plus d'informations, vous pouvez me contacter à **valentin.vgrx@gmail.com**

English

InfoCommerce Project

*Code files are available on my GitHub profile: **ValentinVigouroux**, repository: "InfoCommerce"*

Introduction

In this project, I extracted data from a public website listing financial indicators of French companies using the API of the website **annuaire-entreprise.fr**.

Project Motivation

The goal of this project is to identify which sectors and companies are the most economically dynamic based on aggregated real data for France, and to highlight common characteristics among these companies.
Additionally, the objective was to develop my skills in **data analysis, web scraping, and financial analysis**.

Scraping Justification

An official API is available on **annuaire-entreprise.fr**; however, after requesting an API key, I received no response. I therefore used this scraping method.
I realized that financial data were available in the API request URL made by the page itself and visible in the **Network** tab of the site inspector.

Code Description

This is a **Scrapy spider (s.py)**. From an SQL database containing a list of companies that I built, the scraper primarily extracts the **SIREN** of each company.
It retrieves the content of the API URL request
(https://data.economie.gouv.fr/api/records/1.0/search/?dataset=ratios_inpi_bce&q=siren%3A{sir

en} in JSON format) via the GET method for each company (by constructing the URL containing the SIREN).

The data are organized, preprocessed, and then exported to a CSV file. I then use these data in a **Power BI dashboard**.

Settings in settings.py:

- The crawling speed is set relatively low to avoid overloading **annuaire-entreprise.fr**.
- **Autothrottle**: adjusts the scraper's speed according to site traffic to prevent overload and adopt a more natural behavior (less detectable).
- Random **user-agents** from a list are associated with each request to make bot detection more difficult.
- The robots.txt file is ignored and cookies are disabled → scraping that violates the terms of the robots.txt of **annuaire-entreprise.fr**.

Scraping Warning

Scraping other than via the official API is prohibited by the rules of **annuaire-entreprise.fr**.
This project is intended for **learning and intellectual curiosity**.
I therefore invite you **not to use this code abusively**.
I am sharing this code solely to **showcase my skills** in the job market.
My code is the work of a novice and is request-intensive; after scraping around 10,000 companies, the site will block access for one day.
In case of abuse, you risk being banned from the site or even legal action due to the impact of request overload.

The code extract information on a list of 2,1 M enterprises and thus will not stop automatically : **Perform "Ctrl+c" 2 times to stop the code manually.**

Steps to Run the Code (For Beginners)

I use **Anaconda Navigator**, which makes it easy to install and manage libraries.

1. Open Anaconda (environment manager).
2. Select an environment containing the **Scrapy** and **Protego** libraries (in addition to the default ones).

3. Download the files (spider + Excel file containing the list of company SIRENs) from the InfoCommerce repository.
4. Launch your code editor (**VS Code**) from the environment manager.
5. Open the files in your editor.
6. Open the spider file s.py.
7. In the terminal, navigate to the spiders folder (“cd api/spiders”)
8. Run the program with the command:
9. scrapy runspider s.py -o resultats.csv -s FEED_EXPORT_ENCODING=utf-8-sig -s FEED_EXPORT_FIELDS_DELIMITER=";" -s JOBDIR=crawls/save1

This command will export the data to an Excel file while keeping previously extracted data in memory, allowing you to extract data in multiple sessions if needed.

Warning : The code extract information on a list of 2,1 M enterprises and thus will not stop automatically : **Perform “Ctrl+c” 2 times to stop the code manually.**

Information on the Nature of the Data

In this project, I focused on **purely commercial companies**, that sell and produce goods and services in the market.
I started with a list of companies in France available from **INSEE** (StockUniteLegale_4colonnes.csv), which contains 4 million companies.

I filtered:

- commercial companies (legal category “5_”);
- certain sectors (purely financial companies, insurance, real estate…) for which the collected financial indicators are not relevant to analyze activity.
  (This code is available in the file codeSQLgrosfichier.sql.)

After filtering, the list contains about **2.1 million companies**.
The code directly extracts financial information from this list of SIRENs for each company:

Example:

{

"taux_d_endettement": 0.0,

"credit_fournisseurs_jours": 100.118,

"marge_ebe": 32.125,
"caf_sur_ca": 15.535,
"poids_bfr_exploitation_sur_ca_jours": 129.224,
"marge_brute": 7413504,
"resultat_courant_avant_impots_sur_ca": 18.651,
"date_exercice": "2021-12-31",
"ebit": 1789388,
"rotation_des_stocks_jours": 76.579,
"siren": "310865597",
"chiffre_d_affaires": 9630088,
"confidentiality": "Public",
"resultat_net": 1374249,
"ebe": 3095627,
"type_bilan": "C",
"poids_bfr_exploitation_sur_ca": 35.895,
"ratio_de_liquidite": 144.734,
"credit_clients_jours": 60.701
}

In addition, the data includes sector and workforce information from the source file.

Potential Code Improvements

Several improvements are possible, including:

- Making the code less request-intensive to scrape faster without overloading the site.
- Automatically stopping the code in case of site blocking (response 429).
- Understanding how the spider chooses the order of SIRENs and adapting this method to ensure a **statistically representative** of French economic activity.

- …

I am primarily a data analyst and a beginner in programming.

Contact

For more information, you can contact me at **valentin.vgrx@gmail.com**