Rapport Projet - Ingénierie des Langues

Mohand Oukhemanou et Vincent Vilfeu

29 Mai 2023

Projet de Scraping - Pokédex

Introduction

Dans le cadre de l'EC Ingénierie des Langues, nous sommes amenés à devoir réaliser un projet de scraping afin d'extraire des données textuelles sur un ou plusieurs sites.
Nous avons alors décidé de créer un corpus reposant sur le principe du Pokédex, c'est-à-dire de créer une base de données comportant les pokémons des 4 premières générations ainsi que des informations à leur sujet.
Pour cela nous avons utilisé le site poképédia.fr et les bibliothèques BeautifulSoup et Requests pour la partie scraping, les bibliothèques Pandas et CSV pour celle sur la base de données, les bibliothèques Functools et Operator pour travailler sur les listes et la bibliothèque Re(gex) pour travailler sur les données textuelles.
Dans la suite du rapport, nous expliquerons brièvement le fonctionnement de certaines des bibliothèques que nous avons utilisées, puis nous détaillerons le fonctionnement de notre code pour le scraping des pages.

Bibliothèques

BeautfulSoup & Requests

La bibliothèque BeautifulSoup est une bibliothèque permettant de faire de l'analyse syntaxique de page HTML ou XML.
La bibliothèque Requests est une bibliothèque permettant de réaliser des requêtes HTTP de manière simple.
Ainsi en utilisant les deux ensembles, cela permet de faire du parsing HTML de façon simpliste. Pour cela, il faut également bien analyser le code source de la page en utilisant l'inspecteur (via le raccourci ctrl+shift+i). Dans le but de comprendre sous quelles formes se trouvent nos données dans le code source, pour bien les extraire.

Pandas & CSV

La bibliothèque Pandas est une bibliothèque permettant de manipuler et faire de l'analyse de données. Elle nous sert à stocker les données que l'on extrait des pages web.
La bibliothèque CSV est une bibliothèque permettant de lire et d'écrire des données au format CSV. Elle nous sert à mettre sous la forme de fichier csv (puis xmls) nos données.

Functools & Operator

Les bibliothèques Functools et Operator nous permettent de travailler sur les listes, notamment pour aplatir les listes que nous utilisons. Pour cela on utilise la fonction *reduce()* de Functools et la fonction *concat()* d'Operator.

Re(gex)

La bibliothèque Re est une bibliothèque permettant de travailler sur les expressions régulières. Elle nous sert à modifier les données extraites afin de rendre notre base de données plus lisible.

Nos Fonctions

class Pokemon

La *class Pokemon* contient les fonctions propres à un pokémon (soit à une donnée). Elle contient notre constructeur (avec toutes les caractéristiques que l'on veut récupérer sur les pages web), la fonction d'affichage d'un pokémon (et de ses caractéristiques), la fonction d'affectation des détails d'un pokémon et enfin la fonction d'affectation des stats d'un pokémon.

get_page(url, nom_poke)

La fonction get_page() permet via la fonction request, de récupérer toutes les données textuelles d'une page.
Ces données sont ensuite mise en argument de la fonction BeautifulSoup (en plus de l'argument 'html.parser' permettant d'indiquer le type HTML de notre page) pour créer un objet de type "BeautifulSoup" qui permet de représenter la page sous la forme d'une structure de données imbriquée.

get_infos(table, types, list_of_pokemon))

La fonction get_infos() permet de récupérer les données présentes sur la première page que l'on veut scraper (soit la liste de tous les pokémons).
Cette dernière contient les informations qui nous intéressent, dans un tableau que l'on récupère via son nom de classe.
Il suffit ensuite de récupérer toutes les lignes de ce tableau, puis le contenu de chaque cellule.
Ainsi, on peut ajouter à une liste (celle mise en argument, déclaré vide au préalable) des objets de type Pokémon avec leur numéro, nom français, nom anglais et types.
Le type, apparaissant sur la page par le biais d'une image, ne se récupère pas de la même manière.
On s'intéresse aux objets de type "img" et on récupère le texte contenu dans le tag "alt" de ces dernières.
Pour ne pas prendre trop de temps à la compilation et avoir une base de données trop conséquente, on s'arrête aux pokémons de la 4ème génération (soit 493 pokémons au total) et on évite de récupérer les doublons (le même pokémon avec une forme différente).

get_detail(list_of_pokemon)

Une fois la liste des pokémons obtenue, on s'intéresse aux informations complémentaires propres à chaque pokémon.
La fonction get_detail() nous permet de scraper toutes les données présentes sur la page individuelle des pokémons de notre liste. Pour cela, on va récupérer un à un, chaque pokémon et aller sur sa page individuelle, on prend son nom français (précédemment récupéré) et on l'ajoute à une URL (fonction get_page()) pour scraper sa page.
Les informations qui nous intéressent sur cette page se trouvent également dans un tableau, ainsi on les récupère sur le même principe que précédemment en ajoutant dans le nom de classe le premier type du pokémon ciblé.
Ensuite, pour supprimer les données qui ne nous intéressent pas, on travaille avec des liste via les fonctions reduce() de Functools, concat() de Operator ou encore la fonction pop(). Et pour rendre celles que l'on conserve plus lisibles, en effaçant certains caractères, on utilise la fonction sub() de Regex. Toutes ces données sont alors ajoutées au pokémon en question grâce à la fonction add_attribute() de notre classe.
Une difficulté rencontrée avec cette fonction fut certains pokémons légendaires (numéro 483, 484, 487, 492) qui avait une page individuelle différentes de toutes les autres. Ainsi il fallait adapter le code afin de bien récupérer ce qu'il nous faut, pour cela on a ajouté des conditions avec une façon de faire spécifique à ces pokémons.

get_stat(list_of_pokemon)

Reposant sur le même principe et fonctionnement que la fonction get_detail(), la fonction get_stat() nous permet de scraper toutes les statistiques d'un pokémon de notre liste (également présente sur sa page individuelle).
Ici aussi, elles sont contenues dans un tableau, mais ne voulant que les statistiques de base, on ne récupère que les données de la seconde colonne de ce tableau. Puis on les ajoute, au pokémon via la fonction add_stat() de la classe.
Avec cette fonction, il y a aussi eu une difficulté et cette fois-ci non pas à cause de certains légendaires, mais parce que dans le code source, le tableau des statistiques n'avait pas la même position.
Ainsi son index pouvait avoir des valeurs différentes. Il fallait alors trouver le point commun entre les pages dont le tableau avait le même index, ce fut assez long à déterminer mais à force d'analyser leur code source, on a pu trouver des conditions permettant de réussir à scraper chaque page en évitant ces problèmes d'index.
Mais malgré cette solution, un pokémon continuait de provoquer une erreur, le numéro 413 (un pokémon ayant plusieurs types et dont le tableau des statistiques varie selon ce dernier), il a fallu alors gérer sa situation de manière spécifique.

create_csv_with_data() & convert_csv_to_excel()

Ces deux fonctions, comme leur nom l'indique, permettent de créer un fichier csv et un fichier excel (issus du fichier csv).
La première fonction va ainsi permettre d'obtenir un fichier csv avec les données que l'on a scrapé et sauvegardé dans la liste de pokémons, chaque attribut formant ainsi une colonne. Pour cela, elle va créer le dossier data si ce dernier n'existe pas, puis après avoir récupéré l'emplacement du fichier, va l'ouvrir en écriture afin d'y renseigner nos données.

La seconde va simplement convertir le premier fichier obtenu précédemment en fichier xlsx, grâce à la bibliothèque Pandas et plus particulièrement aux fonctions read_csv() et pd.dataframe.to_excel().

Conclusion

En conclusion, nous avons pu, au travers de notre projet, implémenter une méthode de scraping de plusieurs pages.
Nous avons ainsi pu découvrir et faire usage de la bibliothèque BeautifulSoup, qui rend le scraping très simpliste.
Et si d'ordinaire, scraper une page web peut s'avérer problématique d'un point de vue sécurité (le site bloquant la récupération des données), ce ne fut pas le cas du site que nous avons utilisé.
Pour nous, la difficulté résidait plutôt dans la forme des données dans le code source, qui variait selon la page, provoquant des exceptions à gérer pour notre code.
Pour aller plus loin, nous pourrions essayer de l'améliorer, pour qu'il puisse s'adapter à différents sites (et plus uniquement poképédia.fr) en gérant les particularités qui peuvent exister.