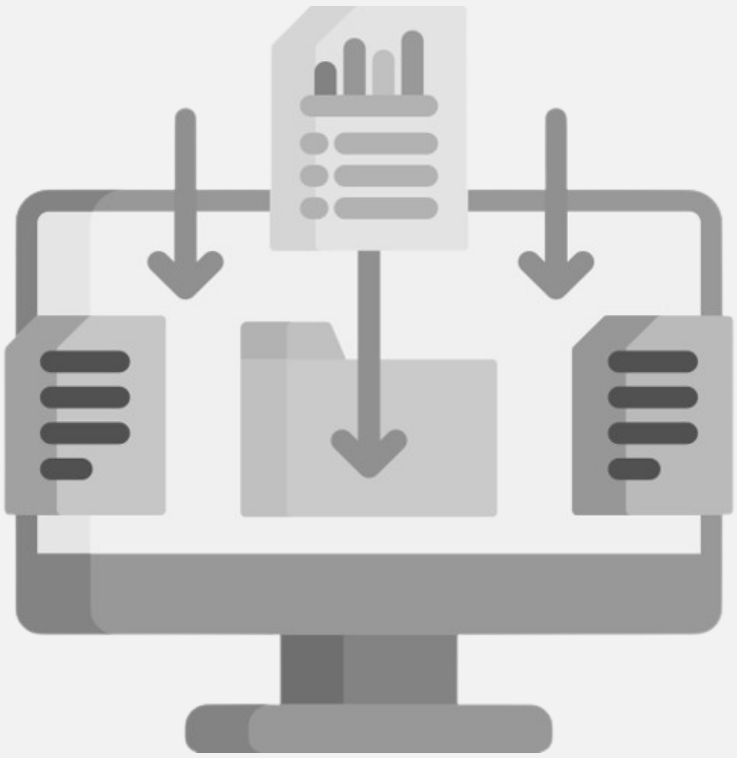
Université Paris Cité

Aymeric MOUTTAPA
Maximilien SOUMAHORO

BUT SD2 – FA VCOD

SAE

Collecte de Données Web



SOMMAIRE

INTRODUCTION

Le cœur de ce projet était d'enrichir la base de données SIRENE, d'abord à partir du fichier StockEtablissement. Ce fichier, bien que riche en informations sur les établissements, nécessite des données supplémentaires. Notre objectif était de compléter ces informations avec les coordonnées géographiques et des détails supplémentaires obtenus grâce à une recherche sur Google Maps.

Pour mener à bien ce projet, nous avons suivi un pseudo-algorithme qui structurait notre démarche de collecte de données.

Il établit que N représente le nombre total de lignes du fichier et le paramètre P est défini comme le nombre de lignes à lire et à traiter à chaque fois, fixé ici à 100 lignes pour l'exemple.

Ce cadre nous guide dans une approche structurée pour aborder le fichier : procéder par segments de 100 lignes à la fois. Pour chaque segment, l'algorithme prévoit d'extraire le numéro SIRET de chaque établissement listé, utiliser ce numéro pour obtenir les coordonnées géographiques via l'API Adresse, et ensuite collecter des données complémentaires via une recherche sur Google Maps.

Les informations obtenues sont ensuite ajoutées au DataFrame initial, ce qui permet d'enrichir la base de données avec les informations et le site web qui peuvent être lier à chaque SIREN. Ce processus est répété itérativement, traitant chaque segment de 100 lignes, jusqu'à ce que l'ensemble du fichier soit parcouru et que la base de données SIRENE soit entièrement mise à jour.

DÉMARCHE

I – Préparation des données :

La première étape du projet consistait à préparer et à nettoyer les données nécessaires à l'analyse. Cette phase impliquait les actions suivantes :

- **Importation des données :** Le fichier StockEtablissement_utf8_1000.csv a été importé en utilisant la bibliothèque pandas de Python. La commande pd.read_csv() a été employée pour lire les premières lignes du fichier, permettant une inspection initiale des données sans avoir à charger le fichier entier en mémoire. Cela était crucial pour gérer l'efficacité de la mémoire, surtout compte tenu de la taille considérable du fichier de données.
- **Sélection des colonnes :** Une fois les données importées, une sélection des colonnes pertinentes a été effectuée. Cette sélection visait à réduire le dataset aux informations essentielles pour la suite du processus, notamment les identifiants SIREN et SIRET, les détails de l'adresse, le code postal, et la commune.
- **Nettoyage et transformation des données :** Les champs de données tels que le numéro de voie ont été nettoyés pour éviter les erreurs. Les valeurs non numériques ont été traitées avec pd.to_numeric() en utilisant errors='coerce' pour convertir les valeurs non convertibles en NaN, puis remplacées par zéro. Cette étape était essentielle pour éviter les erreurs lors des traitements ultérieurs qui nécessitent des formats numériques.
- **Normalisation des données textuelles :** Les abréviations dans les adresses ont été remplacées par des mots complets pour faciliter la concaténation des variables pour créer l'adresse complète. Par exemple, les abréviations "PL", "AV" et "RTE" ont été converties respectivement en "Place", "Avenue" et "Route".

II – Collecte de données géographiques :

La collecte des données géographiques et complémentaires s'articule autour de deux processus principaux : la lecture paginée des données et la génération d'URL pour les requêtes API.

- Lecture paginée des données : Le processus commence par définir la structure de pagination pour traiter le fichier de données par lots. Ceci est réalisé grâce aux variables clés dans ce processus « nrows= » et « skiprows= ».
- Génération des URL pour les requêtes API : La seconde partie du processus implique la génération d'URL pour interroger une API de géocodage afin d'obtenir les coordonnées géographiques pour chaque établissement. Pour chaque ligne lue dans le segment actuel, l'adresse complète est formée à partir des colonnes pertinentes du DataFrame, telles que numeroVoieEtablissement, typeVoieEtablissement, et libelleVoieEtablissement.
  Voici les étapes suivies pour générer les URL :
  -Concaténation des différentes parties de l'adresse pour former une requête complète.
  -Encodage de cette adresse en format URL pour assurer la validité de la requête HTTP.

-Formation de l'URL finale en ajoutant l'adresse encodée à la base de l'URL de l'API de géocodage.
Cette URL est ensuite utilisée pour effectuer une requête GET vers l'API de géocodage, qui retourne les coordonnées géographiques correspondant à l'adresse fournie.

Les longitudes et latitudes ont ensuite été concaténés dans le dataset, ainsi que le Label (utile pour la partie chromedriver).

III – Utilisation de chromedriver pour l'extraction des données supplémentaires:

L'objectif de cette phase était d'utiliser chromedriver avec Selenium pour naviguer sur Google Maps et extraire des informations complémentaires sur les établissements, notamment les informations et les sites web relié à chaque siret. Cette étape nécessitait une interaction automatisée avec un navigateur web pour accéder à des données qui ne sont pas directement accessibles via une API ou un fichier de données.

- **Mise en place de l'environnement :** Le processus a commencé par la configuration de Selenium et chromedriver.
- **Extraction des données :** Pour chaque établissement listé dans le DataFrame sirene, une URL spécifique de Google Maps était construite à partir des données d'adresse. Cette URL était utilisée pour diriger le navigateur automatisé vers la page de Google Maps correspondante à l'établissement. Voici les principales étapes suivies :
  -Construction des URL de recherche : Basé sur les labels extraits précédemment, des URL de recherche Google Maps étaient formées pour pointer vers la localisation spécifique de chaque établissement.
  -Navigation et chargement de la page : chromedriver était utilisé pour ouvrir chaque URL et attendre que la page soit complètement chargée.
  -Extraction des données : Une fois la page chargée, le code HTML de la page était récupéré, et des sélecteurs étaient utilisés pour extraire les informations et les URLs des sites web des établissements. Les données extraites étaient stockées dans des listes correspondantes pour un traitement ultérieur.
  -Gestion des exceptions : Des codes de gestion des erreurs étaient en place pour traiter les cas où les informations n'étaient pas disponibles ou lorsque la page ne répondait pas aux attentes, en insérant des valeurs null pour maintenir l'intégrité de la structure des données.
- **Intégration des données extraites :** Les informations recueillies via chromedriver étaient ensuite intégrées au DataFrame sirene. Chaque établissement dans le DataFrame se voyait attribuer ses coordonnées géographiques, un label détaillé de l'adresse, ainsi que les liens vers les sites web et autres informations pertinentes récupérées.

IV – Écriture des données:

Après avoir enrichi notre DataFrame sirene avec des informations complémentaires (coordonnées géographiques, les détails de localisation, et les données extraites via chromedriver) il fallait sauvegarder ces informations. Voici comment nous avons procédé :

- **Consolidation des données** : Notre DataFrame sirene a été peaufiné pour inclure uniquement les colonnes siren, siret, codePostalEtablissement, libelleCommuneEtablissement, adresse_complete, Longitude, Latitude, Label, Informations, et Site_Web. Cette étape était cruciale pour garantir que toutes les informations pertinentes soient capturées et structurées de manière cohérente.
- **Exportation en format CSV :** Nous avons utilisé la fonction to_csv de Pandas pour écrire le DataFrame finalisé dans un fichier CSV nommé sirene.csv. Pour assurer la compatibilité et la facilité d'accès des données, nous avons opté pour l'encodage UTF-8 et choisi le point-virgule (;) comme séparateur de colonnes.

PROBLÈMES RENCONTRÉS

I – Gestion des boucles :

La structure du projet nécessitait une itération efficace sur les lignes d'un très grand fichier de données pour traiter et collecter des informations supplémentaires. Trouver les bonnes boucles pour parcourir les données, collecter des informations géographiques, et ensuite interroger une API ou un site web, était crucial. Le défi était de gérer ces boucles sans surcharger la mémoire ou sans compromettre les performances, surtout avec un fichier de données volumineux.

II – Gestion des statuts de réponse HTTP :

Lors de la collecte de données à partir de l'API et de Google Maps via des requêtes HTTP, nous avons été confrontés à des défis liés à la gestion des différents statuts de réponse HTTP. Il était crucial de distinguer les réponses réussies des erreurs pour assurer l'intégrité et la fiabilité des données collectées. Les réponses non réussies, comme les erreurs client (400-499) et les erreurs serveur (500-599), pouvaient potentiellement interrompre le processus de collecte de données ou entraîner l'intégration de données incomplètes ou incorrectes. Nous avions donc dû trouver un moyen de contourner les différents statuts non-réussis.

III – Extraction de la longitude et de la latitude :

L'extraction des données de longitude et de latitude pour chaque établissement représentait un défi majeur. Le processus nécessitait une requête précise à l'API de géolocalisation pour chaque adresse, et ce, en tenant compte de la structure et de la précision des données d'adresse disponibles. La difficulté principale résidait dans le fait que les adresses incomplètes ou mal formatées conduisaient à des requêtes infructueuses, ce qui se traduisait par l'absence de coordonnées géographiques dans les réponses de l'API.

IV – Extraction de la longitude et de la latitude :

L'un des défis les plus significatifs de ce projet a été l'utilisation de chromedriver via Selenium pour extraire les informations et les sites web des établissements à partir de Google Maps. Malgré plusieurs tentatives et approches différentes, nous avons constamment rencontré des problèmes, car le chromedriver renvoyait systématiquement des valeurs null.

Le principal problème résidait dans l'extraction des données spécifiques de la page des résultats de Google Maps. Même avec des sélecteurs CSS correctement formulés et des attentes explicites pour s'assurer que les éléments de la page étaient chargés, les requêtes aboutissaient à des valeurs null. Ce comportement suggère que soit les éléments n'étaient pas présents au moment de l'interrogation, soit le contenu était chargé dynamiquement d'une manière que chromedriver ne pouvait pas intercepter efficacement (pas trouver de solution).

V – Traitement des grands fichiers de données :

La manipulation d'un fichier volumineux de 6,84 Go représentait un défi majeur dans ce projet. Le principal obstacle était de traiter et d'analyser efficacement un tel volume de données sans rencontrer de problèmes de performance ou d'épuisement de la mémoire. Il a donc fallu trouver un moyen de lire et traiter le fichier par morceaux (expliquer sur le fichier .ipnyb).

CONCLUSION

Ce projet de compléter la base de données SIRENE a été une aventure pleine d'apprentissages. Nous avons dû jongler entre la lecture d'un gros fichier, la collecte de données géographiques et des détails via Google Maps, et la rencontre avec divers problèmes techniques.

En suivant notre plan étape par étape, nous avons pu avancer méthodiquement. Cela a impliqué de lire le fichier StockEtablissement par petits bouts et d'utiliser ces informations pour trouver des données géographiques et autres détails en ligne. Bien que le processus ait été laborieux, il nous a permis de construire progressivement notre base de données enrichie.

Les défis étaient au rendez-vous, notamment en programmant les bonnes boucles pour traiter les données et en gérant les réponses des requêtes web. La partie la plus coriace a été d'utiliser chromedriver pour récupérer des infos sur Google Maps, où souvent on obtenait des résultats nuls, ce qui nous a pas mal fait tourner en rond (et on n’a toujours pas réussi à récupérer ces informations). La taille du fichier de 6,84 Go a aussi posé problème, nous obligeant à réfléchir à des stratégies pour travailler avec de telles quantités de données sans faire planter notre système.

En résumé, malgré les obstacles, ce projet a été très instructif. On a appris plein de choses sur la gestion des données, le codage pour la collecte d'informations en ligne, et comment s'adapter quand les choses ne se passent pas comme prévu (sauf pour la partie chromedriver ducoup…).