2025

Rapport de projet : Digitalisation des commerces parisiens

Diego CASA BARCENAS
Romain SALMERON
Leo Jean UNITE
IUT Paris – Rives de seine
01/01/2025

Contents

I. Introduction

La digitalisation des commerces est devenue un enjeu majeur pour améliorer leur visibilité et attractivité. Ce projet vise à enrichir la base de données des commerces parisiens en intégrant des informations géographiques et numériques (coordonnées GPS, sites web, options de livraison).

II. Objectifs du projet

Les objectifs de ce projet sont les suivants :

- Enrichir la base de données BCOM2023 avec des coordonnées géographiques précises.
- Ajouter des informations sur les sites web et la disponibilité de services de livraison.
- Faciliter l'analyse et la visualisation des données pour une meilleure prise de décision.

III. Méthodologie

Le projet a été réalisé en plusieurs étapes :

1. **Exploration des données** : Chargement et analyse du fichier de données initial pour identifier les colonnes pertinentes.
2. **Géocodage des adresses** : Utilisation de l'API du gouvernement pour obtenir les coordonnées GPS.
3. **Scraping des sites web** : Extraction automatique d'informations en utilisant Selenium et BeautifulSoup.
4. **Intégration des données** : Fusion des nouvelles données avec le fichier d'origine.

IV. Analyse des données

1. Exploitation initiale

- L’analyse préliminaire du fichier de données *BDCOM2023* a permis de mieux comprendre sa structure et d’identifier les colonnes essentielles pour effectuer le géocodage. Cette étape a été cruciale pour déterminer les champs nécessaires à la construction d’adresses exploitables.

- Les colonnes clés mises en évidence dans cette phase sont les suivantes : **num**, **let**, **typ_voie**, **lib_voie**, et **arro**. Ces champs contiennent les informations principales relatives aux adresses.

2. Préparation des données

Dans le cadre de la préparation des données pour le géocodage, une colonne supplémentaire intitulée **adresse** a été créée. Cette colonne résulte de la concaténation des champs pertinents identifiés précédemment (**num**, **let**, **typ_voie**, **lib_voie**, **arro**). Cette opération vise à générer une représentation textuelle cohérente et complète des adresses, facilitant ainsi leur utilisation dans les processus de géocodage automatisés.

V. Géocodage des adresses

3. Méthode utilisée

L'API de géocodage du gouvernement français (https://api-adresse.data.gouv.fr/) a été utilisée pour convertir les adresses en coordonnées GPS (latitude et longitude).

4. Résultats obtenus

- Le fichier d'adresses préparées a été soumis à l'API.
- Les résultats ont été stockés dans un fichier CSV enrichi.
- Un taux de correspondance élevé a été obtenu, mais des vérifications manuelles ont été effectuées pour corriger certaines erreurs.

VI. Recherche des sites web et services de livraison

5. Outils utilisés

- **Selenium** pour automatiser la navigation sur Google Maps.
- **BeautifulSoup** pour extraire les liens des sites web.
- **Gestion des requêtes** avec des délais pour éviter d'être bloqué par Google.

6. Résultats obtenus

- Extraction des sites web pour la majorité des commerces.
- Identification de la présence ou absence de services de livraison.
- Stockage des informations dans le fichier final.

VII. Intégration et enrichissement des données

Une fois les coordonnées GPS et les sites web extraits, ces données ont été fusionnées avec le dataset initial.

- Colonnes ajoutées : **latitude**, **longitude**, **site**, **livraison**.
- Vérifications pour éviter les doublons et les erreurs d'association.

VIII. Défis rencontrés

Les principales difficultés rencontrées incluent :

- **Qualité des données** : gestion des valeurs manquantes et incohérences.
- **Limitations des APIs** : restrictions d'usage et quotas.
- **Scraping des sites web** : adaptation des scripts pour contourner les protections de Google.
- **Temps de traitement** : optimisation des scripts pour accélérer le processus.
- **Gestion des données :** Duplication des données

IX. Conclusion

Ce projet a permis d'enrichir la base de données des commerces parisiens en fournissant des informations précieuses sur leur localisation et leur présence en ligne. Ces données peuvent être exploitées pour des analyses avancées et aider les commerçants à améliorer leur visibilité.

X. 10. Annexes

7. Structure du projet

a. Etape 1 : Chargement et exploration des données

```
import pandas as pd

# Charger le fichier CSV
file_path = 'data/BDCOM_2023(in).csv'
df = pd.read_csv(file_path, encoding='ISO-8859-1')

# Aperçu des premières lignes
print(df.head())

# Vérification des colonnes disponibles
print(df.columns)

# Vérification des valeurs manquantes
print(df.isnull().sum())
```

✓ 1.2s

```
                X            Y  OBJECTID   c_ord  arro  qua          xbis  \
0  651791.048600003  6.862992e+06       1.0  1311.0   1.0  2.0  651792.345590
1       652152.0612  6.862579e+06       2.0  1464.0   1.0  2.0  652152.061200
2  651430.135700002  6.862714e+06       4.0  1623.0   1.0  3.0  651430.135700
3  651133.490999997  6.862932e+06       6.0  2087.0   1.0  3.0  651130.053214
4  651124.613200001  6.863066e+06       7.0  2157.0   1.0  3.0  651124.613200

          ybis    num  let  ... codact                  ens  bio surf cc_id  \
0  6.862996e+06   25.0  NaN  ...  CB107                  Y'S  0.0  1.0   0.0
1  6.862579e+06    1.0  NaN  ...  CC301        ALAIN AFFLELOU  0.0  1.0   1.0
2  6.862714e+06  196.0  NaN  ...  CH106         ENZA FAMIGLIA  0.0  1.0   0.0
3  6.862939e+06    7.0  NaN  ...  SA202          JULIE BEAUTE  0.0  1.0   0.0
4  6.863066e+06   20.0  NaN  ...  CD201  HOME AUTOUR DU MONDE  0.0  1.0   0.0

  cc_niv    niv47  niv18  niv8  niv2
0    NaN  10301.0  103.0   3.0   1.0
1   -3.0  10403.0  104.0   3.0   1.0
2    NaN  11101.0  111.0   5.0   1.0
3    NaN  10802.0  108.0   4.0   1.0
4    NaN  10502.0  105.0   3.0   1.0

[5 rows x 25 columns]
Index(['X', 'Y', 'OBJECTID', 'c_ord', 'arro', 'qua', 'xbis', 'ybis', 'num',
       'let', 'typ_voie', 'lib_voie', 'seq', 'sit', 'type', 'codact', 'ens',
       'bio', 'surf', 'cc_id', 'cc_niv', 'niv47', 'niv18', 'niv8', 'niv2'],
...
niv18          26
niv8           26
niv2           26
dtype: int64
```

b. Etape 2 : Envoyer les adresse à l'API

```python
import requests

# URL de l'API du gouvernement français pour le géocodage
url = "https://api-adresse.data.gouv.fr/search/csv/"

# Ouvrir le fichier contenant les adresses
with open("data/export_for_search.csv", "rb") as fichier:
    response = requests.post(url, files={"data": fichier})

# Vérifier si la requête a réussi
if response.status_code == 200:
    # Sauvegarder le fichier géocodé
    with open("data/result_geocoded.csv", "wb") as output:
        output.write(response.content)
    print("Géocodage réussi. Fichier sauvegardé sous 'data/result_geocoded.csv'.")
else:
    print("Erreur lors du géocodage :", response.status_code)
```

✓ 1m 36.7s

```
Géocodage réussi. Fichier sauvegardé sous 'data/result_geocoded.csv'.
```

c. Etape 3 : Intégrer les coordonnées dans la dataset initial

```python
# Charger les résultats géocodés
df_geocoded = pd.read_csv("data/result_geocoded.csv", encoding='ISO-8859-1')

# Fusionner avec les données originales sur la colonne "adresse"
df_final = df.merge(df_geocoded[['adresse', 'latitude', 'longitude']], on='adresse', how='left')

# Supprimer les doublons
df_final = df_final.drop_duplicates()

# Sauvegarder le fichier final enrichi
df_final.to_csv("data/BCOM2023_enriched.csv", index=False, encoding='ISO-8859-1')
print("Fichier enrichi avec coordonnées sauvegardé.")

df_final
```

✓ 1.6s

d. Etape 4 : Recherche des sites web des commerces

Code test de recherche & livraison

```
# Liste pour stocker les résultats
site = []
livraison = []

# Limiter le DataFrame aux n premières lignes
df_subset = df.head(5)

# Boucle pour parcourir les données et rechercher les sites web
for i in range(df_subset.shape[0]):
    place_info = f(df["num"][i]) + f(df["let"][i]) + f(df["typ_voie"][i]) + f(df["lib_voie"][i], True)
    comp_url = "/@" + str(df["latitude"][i]) + "," + str(df["longitude"][i])
    url = base_url + place_info + comp_url
    driver.get(url)
    html = driver.page_source
    soup = BeautifulSoup(html, "html.parser")

    # Extraction du site web
    results = soup.select("a[aria-label*='Site Web']")

    # Si un résultat est trouvé, ajoutez le lien, sinon ajoutez None
    if results:
        href = results[0]["href"]
    else:
        href = None

    site.append(href)  # Ajouter le lien du site web ou None

    # Recherche de la mention "livraison" (à adapter selon la structure des sites)
    if "livraison" in html.lower() or "deliver" in html.lower() or "uber" in html.lower():
        livraison.append("Oui")
    else:
        livraison.append("Non")

    time.sleep(1)  # Respecter les délais pour éviter d'être bloqué

# Ajouter les résultats dans le sous-ensemble de DataFrame
df_subset["site"] = site
df_subset["livraison"] = livraison

# Sauvegarder le fichier avec les nouveaux résultats
df_subset.to_csv("data/BDCOM_2023_avec_site_et_livraison_test.csv", index=False)

print("Fichier sauvegardé pour les n premières adresses.")
```