

Ingénierie de Données / Transformation Digitale et Intelligence Artificielle

Semestre 5

Année universitaire 2025/2026

Module : Web Marketing & CRM

TP N° 02

Pr : Sara OUALD CHAIB

Objectif

Ces travaux pratiques vous permettent d'appliquer vos compétences en data engineering et IA au domaine du SEO. Vous allez collecter, analyser et visualiser des données web réels pour optimiser le référencement naturel.

Partie - Atelier d'innovation

Exercice 1 :

ETUDE DE CAS : Jumia Maroc (Approche Data-Driven du SEO)

Contexte :

Jumia est la plus grande plateforme e-commerce en Afrique avec des milliers de produits en ligne. Suite à une récente mise à jour de leur plateforme, l'équipe marketing a constaté une baisse significative du trafic organique sur certaines catégories de produits.

Une analyse préliminaire révèle plusieurs problèmes SEO récurrents sur les pages produits : titres trop longs ou manquants, descriptions pauvres et parfois absentes, images sans attribut ALT, contenu insuffisant, et structure de balises Hn non conforme.

Problématique :

Comment identifier et quantifier automatiquement les défauts SEO On-Page à travers un échantillon représentatif de 30 à 100 pages produits de Jumia, afin de prioriser les corrections et améliorer le positionnement organique ?

Mission :

Développer un outil Python d'audit SEO automatisé capable de scraper et analyser maximum 100 pages produits Jumia, d'identifier les non-conformités SEO selon des critères prédefinis, et de générer un rapport actionnable avec des visualisations pour faciliter la prise de décision.

Site cible : <https://www.jumia.ma> (catégorie Electronique)

Conseil : Créez un environnement virtuel Python dédié pour ce TP. Installez les dépendances avec : pip install beautifulsoup4 selenium pandas matplotlib plotly requests lxml

1. Données à extraire

Pour chaque page produit Jumia, votre scraper doit extraire :

- **URL complète** : L'adresse de la page analysée
- **Balise Title** : Contenu et longueur en caractères
- **Meta Description** : Contenu et longueur (si présente)
- **Balises H1** : Nombre et contenu de chaque h1
- **Balises H2** : Nombre total de h2
- **Images** : Nombre total d'images et nombre d'images sans attribut ALT
- **Contenu textuel** : Nombre de mots dans la description du produit
- **Prix** : Prix du produit
- **Catégorie** : Catégorie du produit (sous-catégorie de la catégorie Electronique)

2. Règles de validation SEO pour 100 pages :

Elément SEO	Critere optimal	Statut ERREUR si
Balise Title	Entre 50 et 60 caractères	< 40 ou > 70 caractères
Meta Description	Entre 150 et 160 caractères	Absente ou < 120 caractères
Balise H1	Exactement 1 H1 par page	0 H1 ou plus de 1 H1
Images ALT	100% des images avec ALT	Plus de 30% sans ALT
Contenu produit	Plus de 200 mots	Moins de 150 mots
Structure H2	Au moins 2 H2	0 ou 1 seul H2

3. Architecture du code attendue

Votre solution doit suivre une architecture orientee objet avec les composants suivants :

```
# Structure de fichiers recommandee
jumia_seo_scraper/
    ├── scraper.py
    ├── validator.py
    ├── analyzer.py
    ├── main.py
    ├── requirements.txt
    └── README.md
```

4. Livrables

1. **Code source complet** : Tous les fichiers Python avec commentaires explicatifs (maximum 300 lignes au total)
2. **Rapport CSV** : Fichier jumia_audit_seo.csv contenant toutes les pages analysees avec leurs métriques et statuts
3. **Tableau de bord visuel** : Fichier jumia_dashboard.png avec au moins 4 graphiques : distribution des erreurs, top pages problématiques, répartition par type d'erreur, évolution potentielle
4. **Rapport de synthèse** : Document jumia_executive_summary.txt (200-300 mots) résumant les problèmes identifiés et les actions prioritaires
5. **Documentation** : README.md expliquant l'installation, l'utilisation et les résultats obtenus

Contraintes et bonnes pratiques

- **Respect des serveurs** : Ajouter un délai de 2 secondes entre chaque requête (time.sleep(2))
- **Gestion des erreurs** : Utiliser try/except pour gérer les pages inaccessibles ou mal formatées
- **User-Agent** : Définir un User-Agent explicite pour identifier votre scraper
- **Logs** : Afficher la progression du scraping (ex: Page 15/100 traitée)
- **Code propre** : Respecter les conventions PEP 8, commenter les parties complexes

Astuce : Commencez par tester votre scraper sur 5 pages seulement pour valider la logique avant de passer à 100 pages. Utilisez des print() pour debugger le parsing HTML.

**Important : Ne partagez jamais publiquement du code de scraping agressif.
Respectez toujours le fichier robots.txt et les conditions d'utilisation des sites web.**

Exercice 2 :

ÉTUDE DE CAS : NewsHub Media

Contexte :

NewsHub est un média en ligne avec 50 000 articles. Google crawle environ 10 000 pages/jour mais n'indexe que 40% des nouvelles pages. Le crawl budget est mal utilisé : Googlebot perd du temps sur des URLs obsolètes et des pages de pagination.

Mission :

Analyser 1 mois de logs Apache pour identifier les gaspillages de crawl budget et proposer des optimisations (robots.txt, redirections, sitemap priorité).

Données fournies : Fichier access.log (format Apache, 500k lignes)

- Parser les logs Apache/Nginx avec regex ou pandas
- Filtrer les visites de Googlebot (User-Agent)
- Identifier les URLs les plus crawlées vs les moins crawlées
- Détecter les problèmes : 404, redirections en chaîne, pages obsolètes
- Créer un rapport d'optimisation avec KPIs

Exemple : Format des logs Apache

```
# Ligne de log
66.249.66.1 - - [15/Jan/2025:10:23:45 +0000] "GET /article/seo-2025 HTTP/1.1"
200 15234 "-" "Mozilla/5.0 (compatible; Googlebot/2.1)"
```

Format : IP - - [Date] "Méthode URL Protocole" Code_HTTP Taille Referrer User-Agent

Analyses à réaliser

Analyse	Indicateur
Distribution temporelle	Nombre de crawls par jour/heure (identifier pics)
Top URLs crawlées	20 URLs les plus crawlées (vérifier pertinence)
Erreurs 4xx/5xx	% de crawls sur pages d'erreur (gaspillage)
Profondeur de crawl	Distribution par profondeur d'URL (/lvl1/lvl2/...)
Pages obsolètes	URLs avec date ancienne ou pattern /archive/

Livrables

- **log_analyzer.py** : Script d'analyse des logs
- **crawl_report.csv** : Données agrégées (URL, crawl_count, status_codes, avg_response_time)
- **dashboard.html** : Dashboard interactif Plotly avec 5 graphiques minimum
- **recommendations.md** : Rapport écrit : problèmes identifiés + solutions techniques (robots.txt, redirections, etc.)