Rapport de Laboratoire n°1 – Web Mining

Anthony Atallah, Bérisha Véton, Kouma Assagnon

28.02.2025

Table des matières

1 Introduction

Ce laboratoire avait pour objectif d'implémenter un système complet de Web Mining en quatre étapes principales :

- **Crawling** : Développement d'un crawler en Python utilisant la librairie Scrapy pour parcourir un site web choisi, en récupérant à la fois du contenu textuel et des données structurées (par exemple, en JSON-LD selon *schema.org*).
- **Indexation** : Utilisation d'ElasticSearch pour créer un index unique regroupant toutes les pages crawlées, avec une configuration précise des champs à indexer et l'utilisation d'analysateurs adaptés (notamment pour la langue française).
- **Recherche** : Implémentation d'une fonction de recherche qui exploite l'index créé, en privilégiant certains champs (comme le titre ou les éléments mis en avant) grâce à des requêtes boostées.
- **Questions Théoriques** : Réponses aux questions sur l'indexation multilingue et la recherche floue (fuzzy query) dans ElasticSearch.

2 Description et Implémentation

2.1 Crawler

Pour la partie crawling, nous avons développé un *spider* avec Scrapy afin de parcourir le site choisi.

Points clés :

- **Choix du site et des données** : Nous avons sélectionné un site présentant un contenu riche et structuré : la page Wikipédia sur la Transformation de Fourier.
- **Extraction des données** : Nous avons exploité les *Selectors* de Scrapy pour extraire les éléments suivants :
  - Le titre de la page.
  - Les en-têtes (*headings*) importants.
- **Gestion de la navigation** : Le crawler suit les liens internes pour explorer l'ensemble du site, tout en évitant de se faire « blacklister ».

2.2 Indexation

La seconde étape consiste à indexer les données extraites dans ElasticSearch.

Configuration et choix techniques :

- **Environnement** : Nous avons utilisé Docker avec un fichier `docker-compose.yml` pour lancer un conteneur ElasticSearch et Kibana, facilitant ainsi la visualisation et l'administration de l'index.
- **Mapping de l'index** : Nous avons créé un index nommé `wiki_headings` avec la configuration suivante :
  - **Champs indexés** :
    - `url` (type `keyword`, pour des recherches exactes)
    - `title` (type `text`, avec un analyiseur français personnalisé et boosté lors des recherches)
    - `headings` (type `text`, également indexé avec l'analyiseur français)

— **Analyseur** : Utilisation de l'analyiseur `french` avec des stopwords adaptés pour améliorer la pertinence des recherches en langue française.

— **Script d'indexation indexdata.py** : Un script Python (`index_data.py`) a été développé pour :

— Lire le fichier JSON contenant les données crawlées.

— Créer l'index si nécessaire et y ajouter les documents via une opération `bulk` avec la librairie `elasticsearch`.

2.3 Recherche

La troisième étape consiste à implémenter une fonctionnalité de recherche sur l'index.

Implémentation :

— **Script de recherche** : Un second script Python (`search.py`) permet de se connecter à ElasticSearch et d'exécuter des requêtes.

— **Requête Boostée** : Pour privilégier les champs essentiels, nous avons utilisé une requête `multi_match` qui booste le champ `title` par rapport au champ `headings`. Exemple de requête :

```
GET /wiki_headings/_search
{
  "query": {
    "multi_match": {
      "query": "transformation de Fourier",
      "fields": ["title^3", "headings"]
    }
  }
}
```

Cette requête permet d'obtenir des résultats où les occurrences dans le titre influencent davantage le score de pertinence.

— **Validation** : La fonctionnalité a été validée via Kibana (Dev Tools) et par l'exécution locale du script de recherche. Les résultats retournés incluent le score de chaque document, ce qui permet d'ajuster la pondération des champs si nécessaire.

```
Score: 12.007555
Document: {'headings': ['Transformation de Fourier pour les fonctions intégrables', 'Transformation de
--------------------------
Score: 10.026527
Document: {'headings': ['Définition', 'Transformée de Fourier', 'Applications', 'Note', 'Voir aussi']}
--------------------------
```

FIGURE 1 – Score attribué au document par rapport au mot recherché.

2.4 Questions Théoriques

2.4.1 Stratégies d'Indexation Multilingue

Chaque page étant dans une seule langue mais le corpus comportant plusieurs langues, plusieurs stratégies peuvent être envisagées :

— **Utiliser des analyseurs spécifiques par langue** : Créer des mappings qui appliquent un analyiseur dédié pour chaque langue (par exemple, `french`, `english`, etc.).
Avantages : Meilleure précision dans l'analyse linguistique (tokenisation, suppression des stopwords, stemming).
Inconvénients : Complexité de configuration et maintenance, surtout si le nombre de langues est élevé.
— **Séparer les pages dans des indices distincts** : Créer un indice par langue.
Avantages : Simplicité dans l'application de l'analyseur adapté.
Inconvénients : Complexité lors des recherches multi-langues, car il faut interroger plusieurs indices.
— **Utiliser des champs multilingues dans un même document** : Stocker la même information avec différents analyseurs dans des sous-champs dédiés.
Avantages : Permet des recherches précises sans multiplier les indices.
Inconvénients : Augmentation de la taille de l'index et complexité dans la requête.

2.4.2 Recherche Floue (Fuzzy Query)

ElasticSearch permet de réaliser des recherches floues en utilisant une distance de Levenshtein, qui mesure le nombre de modifications nécessaires pour transformer un terme en un autre.

— **Principe de la fuzzy query** : Elle permet d'obtenir des correspondances même si le terme recherché comporte des erreurs typographiques ou des variations orthographiques.
— **Cas des prénoms avec multiples variations** : Par exemple, pour le prénom « Caitlin » et ses nombreuses variantes (Caitilyn, Catelin, etc.), une fuzzy query peut couvrir ces variations.
— **Performance et alternatives** :
  — *Inconvénients* : Une utilisation extensive peut dégrader les performances sur un large corpus.
  — *Alternatives* :
    — **Indexation de synonymes** : Créer un dictionnaire de synonymes pour normaliser les termes lors de l'indexation.
    — **Prétraitement des données** : Standardiser les orthographes en amont pour limiter l'usage de la fuzzy query.

3 Conclusion

Nous avons implémenté un système complet de Web Mining comprenant :

— **Le crawling** : Extraction efficace de données pertinentes d'un site web choisi, incluant des données structurées.
— **L'indexation** : Mise en place d'un index ElasticSearch avec des mappings adaptés et un analyiseur français pour garantir une recherche précise.
— **La recherche** : Développement d'une fonction de recherche exploitant des requêtes boostées pour améliorer la pertinence des résultats.
— **Les aspects théoriques** : Discussion sur les stratégies d'indexation multilingue et sur l'utilisation des requêtes floues, en soulignant les avantages et limites de chaque approche.

Ce laboratoire nous a permis de comprendre et de mettre en œuvre les concepts clés du Web Mining et d'acquérir une expérience pratique avec des outils tels que Scrapy et ElasticSearch. Pour aller plus loin, il serait intéressant d'envisager une intégration plus poussée (par exemple, indexer directement depuis le crawler via une pipeline) et d'explorer d'autres techniques de normalisation pour améliorer la recherche sur un corpus multilingue.