24 mars 2022

Rapport Laboratoire n°1

MA-WEM – Printemps 2022

Clercq Maxime Demeusy Jean Riondet Guillaume

# Fonctionnement code source

Le code de notre projet se découpe en 3 parties :

* Le module **crawler** qui s’occupe du scraping des pages web et d’enregistrer le résultat dans un fichier JSON.
* Le module **queries** qui permet de faire des querie sur ElasticSearch en choisissant les champs sur lesquels la querie s’exécutera, et le texte à chercher.
* Le module **elasticsearch** qui sert à transformer les données produites par le crawler en structure de données pour ElasticSearch.

**Module crawler :**

Ce module s’occupe de récupérer la page web renseigner (ici <https://www.huffpost.com/news> ) puis la parse pour ne conserver que le texte correspondant au champs CSS renseigner dans le parser. Le processus se répète ensuite de page en page, jusqu’à avoir parcouru toutes les pages de la section news dans notre exemple.

**Module elasticsearch :**

Ce module s’occupe de transférer le résultat de la recherche du crawler dans une structure de donnée qu’ElacticSearch comprend avec, un id, un titre, une description et un auteur défini dans le fichier *article.py*

Une image contenant texte

Description générée automatiquement

Figure 1 Transfert et indexation du crawler vers ElasticSearch

**Module queries :**

Ce module est un wrapper pour faire des requêtes sur ElasticSearch. Le fichier *article.py* construit la structure de données utilisé dans ElasticSearch comme suit :

Une image contenant texte

Description générée automatiquement

Figure 2 article.py

Chaque Article est donc constitué d’un titre, d’une description, d’un auteur et d’un tag (associé au titre de l’article).

Pour l’utiliser, il suffit de lancer dans un terminal la commande :

python -m queries -f <field> -t <text>

En remplaçant <field> par « author », « title » ou « description ». Un exemple est mis à disposition dans la figure 3.

* Dans votre rapport, détaillez la syntaxe utilisée par ElasticSearch pour les faire des querys simples. Comment fait-on pour rechercher uniquement dans certains champs ? Comment fait-on pour « booster » certains champs par rapport à d’autres.

Pour « booster » certains champs par rapport à d’autres, il faut attribuer un score plus élevé à ces champs. Ainsi, s’ils apparaissent dans la requête, ils seront mis plus en avant.

Voici comment faire une querie simple dans ElasticSearch depuis notre système :

Une image contenant texte

Description générée automatiquement

Figure 3 Recherche tous les articles contenant le mot "American" dans leur titre

Avec python il faut lancer le module **queries** en lui donnant les paramètres suivants :

-t : Le texte à rechercher dans la querie

-f : les champs sur lequel la querie doit s’exécuter

Dans l’exemple ci-dessus, nous avons fait une recherche sur le champ « title » avec comme mot clé « American ».

Le résultat de la requête est affiché en fonction du score de retourné par ordre descendant.

# Questions théoriques

* Veuillez expliquer quelle(s) stratégie(s) il faut adopter pour indexer des pages dans plusieurs langues (chaque page est composée d’une seule langue, mais le corpus comporte des pages dans plusieurs langues). A quoi faut-il faire particulièrement attention, quels sont les avantages et inconvénients de celle(s)-ci ? Veuillez expliquer la(les) démarche(s) que vous proposez.

Une indexation par mot clé dans une langue unique peut être réalisée afin d’avoir une base de comparaison dans une seule et même langue. Pour que cette technique soit réalisable il faut s’assurer que les mots-clés fournis par les pages sont en anglais et le cas échéant les traduire mais un souci de précision de la traduction peut apparaitre alors. Dans le cas où toutes les pages fournissent des mots clés dans une seule et même langue alors la recherche peut devenir très intéressante de par la pluralité des langues au final disponible dans le résultat de la recherche.

* *ElasticSearch* permet par défaut de faire de la recherche floue (fuzzy query). Veuillez expliquer de quoi il s’agit et comment *ElasticSearch* l’a implémenté. Certains prénoms peuvent avoir beaucoup de variation orthographiques (par exemple Caitlin : Caitilin, Caitlen, Caitlinn, Caitlyn, Caitlyne, Caitlynn, Cateline, Catelinn, Catelyn, Catelynn, Catlain, Catlin, Catline, Catlyn, Catlynn, Kaitlin, Kaitlinn, Kaitlyn, Kaitlynn, Katelin, Katelyn, Katelynn, etc). Est-il possible d’utiliser, tout en gardant une bonne performance, la recherche floue mise à disposition par *ElasticSearch* pour faire une recherche prenant en compte de telles variations ? Sinon quelle(s) alternative(s) voyez-vous, veuillez justifier votre réponse.

La fuzzy query est une recherche permettant de prendre en compte un écart d’orthographe dans les termes utilisé pour la recherche. Elle implémente la distance de Levenshtein qui consiste à calculer la différence entre deux séquences de string afin d’identifier avec le résultat et à l’aide d’un seuil si deux orthographe d’un mot sont proche ou non.

La fuzzy query risque de détériorer la performance de la recherche étant donné que des cas alternatifs des termes de recherches sont pris en compte, il convient de fixer un seuil (fuziness) de différence pas trop grand afin de réaliser un compromis entre la performance de base sans fuzzy query et la prise en compte d’un grand nombre d’alternative possible à l’orthographe de la query. Une longueur de prefix peut être verrouillée afin d’essayer de conserver le radical par exemple d’un mot ou encore un nombre global de cas pris en compte peut également être envisagé (max\_expansions).

Pour répondre formellement, oui c’est possible de garder une bonne performance mais il faut réellement que la fuzzy query soit paramétrée correctement.

Ref : <https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-fuzzy-query.html>