TME 5 - Outils pour la RI

Il existe de nombreux outils pour faire de la recherche d'information : Lucène, Indri, Terrier, Whoosh, Elastic Search (basé sur Lucène), ... Juste pour vous donner un aperçu, nous allons explorer deux d'entre eux.

Exercice 1 – Whoosh

Whoosh (https://whoosh.readthedocs.io) est une librairie pur-python pour la RI qui présente de nombreuses fonctionnalités.

1. Installer la librairie (https://pypi.org/project/Whoosh/).
2. Parcourir la documentation et explorer l'ensemble des fonctionnalités de la librairie.
3. Indexer et interroger la collection cacmShort.txt

Exercice 2 – ElasticSearch

ElasticSearch présente l'avantage de supporter des collections importantes car il est basé sur des traitements distribués.

1. Télécharger Elastic Search (https://www.elastic.co/fr/downloads/elasticsearch et Kibana (https://www.elastic.co/downloads/kibana)
2. Décompresser et lancer ElasticSearch : `./bin/elasticsearch`. documentation : https://www.elastic.co/guide/en/elasticsearch/reference/current/starting-elasticsearch.html
3. Décompresser et lancer Kibana : `./bin/kibana`. Documentation : https://www.elastic.co/guide/en/kibana/6.7/start-stop.html
4. Parcourir et exécuter certains éléments de la documentation d'ElasticSearch (principalement DocumentAPI, SearchAPI) afin de comprendre l'indexation, la recherche, ... https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html.
   Pour information, ElasticSearch est généralement lancé sur Kibana (http://localhost:5601) qui est une interface de visualisation et qui permet de faire des requêtes API REST. En cliquant sur "View in console" des différents exemples, vous pourrez ensuite exécuter la requête sur Kibana.
5. Les pages principales des "How-tos" :
   — Création des index https://www.elastic.co/guide/en/elasticsearch/reference/current/indices.html
   — Documents https://www.elastic.co/guide/en/elasticsearch/reference/current/docs.html
   — Recherches https://www.elastic.co/guide/en/elasticsearch/reference/7.0/query-dsl-query-str
   html
   — Egalement : https://www.elastic.co/guide/en/elasticsearch/reference/current/cat.html

Il est également possible d'interfacer ElasticSearch avec Python. Très pratique pour faire des expérimentations et surtout si on souhaite parser des grandes collections de documents et les indexer. Ci-dessous, quelques éléments de code :

```
1### Instalation
2pip3 install elasticsearch
3
4### Usage
5###Once you started ElasticSearch, you can instantiate an ElasticSearch object :
6import elasticsearch as es
7engine = es.Elasticsearch()
8
9
```

```
###To create an index :
engine.indices.create(f"{index_name}", body=settings)

###Here is a complete example
index_name = "test-toto"
b = 0.5
k1 = 1
settings = {"settings": {
                "number_of_shards": 1,
                "index": {
                    "similarity": {
                        "default": {
                            "type":"BM25",
                            "b": b,
                            "k1": k1}
                        }
                    },},
                "mappings": {
                    "trec": {
                        "properties": {
                            "title": {
                                "type": "text",
                                "index": "false"
                            },
                            "text": {
                                "type": "text",
                                "analyzer": "english"
                                }
                            }
                        }
                    }
                }
engine.indices.create(f"{index_name}", body=settings)


###Using bulk to index multiple documents:
import elasticsearch.helpers as helpers
def bulking(texts, titles):
    bulk_data = []
    for i, (txt, title) in enumerate(zip(texts, titles)):
        data_dict = {
                    '_index': 'test-toto',
                    '_type': 'trec',
                    '_id': str(i),
                    '_source': {
                        "text": txt,
                        "title": title
                    }
                }
        bulk_data.append(data_dict)
    return bulk_data

texts = ['toto is now gone',
         'toto has left the place',
         "toto went to his parent's place",
         "toto's place is quite far away"]

titles = ["Toto's life",
          "Toto leaving",
```

```
69            "Toto's trip",
70            "Toto is far"]
71
72 helpers.bulk(engine, bulking(texts, titles))
73
74###Search example :
75 engine.search("test-bm25", "trec", {"query": {"match": {"text": "toto place"}}})
```

Une autre façon de faire est d'utiliser le shell : https://openclassrooms.com/fr/courses/4462426-maitrisez-les-4474691-etudiez-le-fonctionnement-d-elasticsearch

A vous de jouer!!!!