Département d'Informatique
Cours de TALN
Demo sur la RI avec Solr / PySolr

Université du Québec à Montréal

Par

Ali MIJIYAWA **Prof: Fatiha Sadat**

November 28, 2024

Table des matières:

- Redefintion d'un fieldtype avec curl
- Indexation des documents avec post
- Indexation des documents avec python
- Requête et calcul des scores
- Résumé sur les filtres
- Exemple de recherche d'informations selon la similarity BM25 sans prétraitement

Solr Versus PySolr

Université du Québec à Montréal

Qu'est ce que Solr ?

- Solr (prononcé comme le mot solar en anglais) est une plateforme logicielle de moteur de recherche s'appuyant sur la bibliothèque de recherche Lucene, créée par la Fondation Apache et distribuée et conçue sous licence libre.
- C'est un projet de la fondation Apache mis à disposition sous licence Apache.
  Solr utilise le langage Java et, jusqu'à la version 5.0, est exécuté par un conteneur de servlets, comme Tomcat, avant de devenir un standalone Java. Il communique avec le client à l'aide d'une interface de programmation en XML et JSON, généralement via le protocole HTTP.

Qu'est-ce que PySolr ?

- pysolr est un client Python léger pour Apache Solr . Il fournit une interface qui interroge le serveur et renvoie des résultats en fonction de la requête.
- Dans la suite, on fera la demo de l'installation et de l'utilisation de Solr (version 9.7.0) et de PySolr.

Installation de PySolr

Université du Québec à Montréal

Pre-requis de l'installation de Solr 9.7.0

- Système d'exploitation : Linux, macOS ou Windows (avec un environnement compatible)
- Python : Version >= 3.11
- Java Development Kit (JDK) : Version >= 11

Editeur de code

- Pycharm community

Conseils généraux pour l'installation de Solr & PySolr

- Vérifier la compatibilité des versions : Assurez-vous que les versions de Java, Python, solr et PySolr sont compatibles entre elles.
- Définir correctement les variables d'environnement :
  — JAVA_HOME : Chemin vers votre installation JDK.

Installation de Solr: version 9.7.0 (Dernière version)

- Installer JDK (de préférence une version >= 11)
- Télécharger Solr à partir du lien:
  `https://solr.apache.org/downloads.html`
  (Télécharger le Binary releases: (solr-9.7.0.tgz))
- Extraire le dossier solr-9.7.0 et le mettre à un emplacement souhaité (de préférence dans un project de pycharm)
- Entrer dans le dossier solr-9.7.0 à travers la console en se servant de la commande "CD"

Installation de JDK

- Verifier s'il y a déjà un JDK installé et sa version en faisant la commande: "java -version"
- Installer JDK (de préférence une version $>= 11$) dans le cas ou la version déjà existante sur votre ordinateur < 11
- Comment definir correctement les variables d'environnements Java ?

Définition de la variable JAVA sur Windows

Pour définir la variable JAVA, exécutez les commandes suivantes dans powershell :

```
[System.Environment]::SetEnvironmentVariable("JAVA_HOME",
"C:\path\to\jdk", "Machine")
```

Définition de la variable JAVA sur Linux

```
echo 'export
JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64' >>
~/.bashrc
echo 'export PATH=$JAVA_HOME/bin:$PATH' >> ~/.bashrc
source ~/.bashrc
```

Installation de PySolr: version 3.10.0

- Faire la commande: pip install pysolr ou
- Installer dans la section paramètre de Pycharm

Démarrage de Solr

- Après avoir procéder à ces installations on pourra démarrer Solr avec succès en mode cloud sur le port 8993 en faisant la commande:
  "bin/solr start -c -p 8993"
- Une fois que le démarrage de Solr est fait on pourra accéder à l'interface de solr via le lien: `http://localhost:8993/solr`

Création des collections et configuartion de Solr

Création des collections

- Après avoir démarrer Solr pour la première fois on procède à la création des collections. (Une collection est une unité d'indexation en solr cloud)
- La commande pour créer une collection du nom "BM25_sans_pret" par exemple est:
  "bin/solr create -c BM25_sans_pret -s 2 -rf 2"
- -s 2: signifie 2 shards (un shard est une subdivision d'une collection)
- -rf 2: signifie 2 replications (permet de créer 2 copies de chaque collection, ce qui permet d'assurer une disponibilité des données en cas de pannes)

Configuration de la Similarité avec la commande curl

--- Configurer les paramètres de similarités (Cette commande permet de définir une similarité par défaut pour toute la collection)

--- Listes des similarity disponibles :
https://solr.apache.org/guide/solr/latest/indexing-guide/schema-elements.html#similarity

```
curl -X POST -H 'Content-type:application/json' --data-binary '{
"set-similarity": {
 "class": "solr.BM25SimilarityFactory",
        "params": {
                "k1": 1.5,
                "b": 0.75
                }
                }
 }' "http://localhost:8983/solr/BM25_sans_pret/schema"
```

NB : Il est possible de spécifier une similarité pour un champ en insérant : "similarity": { "class": "solr.nom_similarity" } dans la configuration de ce field.

Configuration d'un nouveau champ avec la commande curl

--- Création d'un champ (field)

```
curl -X POST -H 'Content-type:application/json' --data-binary '{
  "add-field": {
    "name": "tags",
    "type": "text_general",
    "multiValued": true,
    "indexed": true,
    "stored": true
  }
}' "http://localhost:8983/solr/nom_de_votre_collection/schema"
```

Configuration d'un nouveau champ avec la commande curl

- Lister les champs avec la commande:

```
curl "http://localhost:8983/solr/my_collection/schema/fields"
```

Suppression d'un champ avec curl

- Commande de suppression d'un champ avec curl:

```
curl -X POST -H 'Content-type:application/json' --data-
binary '{
  "delete-field": {
    "name": "my_field"
  }
}' "http://localhost:8983/solr/my_collection/schema"
```

Redefintion d'un fieldtype avec curl

```
curl -X POST -H 'Content-type:application/json' --data-binary '{
 "add-field-type": {
  "name": "text_general",
  "class": "solr.TextField",
  "positionIncrementGap": "100",
  "similarity": { "class": "solr.BM25SimilarityFactory" },
  "analyzer": {
   "tokenizer": { "class": "solr.StandardTokenizerFactory" },
   "filters": [
    { "class": "solr.LowerCaseFilterFactory" },
    { "class": "solr.StopFilterFactory", "words": "stopwords.txt", "ignoreCase": true },
    { "class": "solr.SynonymFilterFactory", "synonyms": "synonyms.txt",
"ignoreCase": true, "expand": true },
    { "class": "solr.PorterStemFilterFactory" }
   ]
  },
  "queryAnalyzer": {
   "tokenizer": { "class": "solr.StandardTokenizerFactory" },
   "filters": [
    { "class": "solr.LowerCaseFilterFactory" },
    { "class": "solr.StopFilterFactory", "words": "stopwords.txt", "ignoreCase": true },
    { "class": "solr.SynonymFilterFactory", "synonyms": "synonyms.txt",
"ignoreCase": true, "expand": true },
    { "class": "solr.PorterStemFilterFactory" }
   ]
  }
 }
}' "http://localhost:8983/solr/nom_de_votre_collection/schema"
```

NB : Le stemmer plus proche de la lemmatisation est : KstemFilterFactory (Dans solr il n' y a pas de classe directe pour la lemmatisation)

Exemple d'indexation des documents avec post

post -c BM25_sans_pret json_docs/*.json (fichiers json)

post -c BM25_sans_pret json_docs/*.xml (fichiers xml)

Exemple d'indexation des documents avec python

Indexation avec python :

```
import pysolr
import json
import os

# Configurer la connexion Solr
solr_url = 'http://localhost:8983/solr/BM25_sans_pret'
solr = pysolr.Solr(solr_url, always_commit=True)

# Dossier contenant les fichiers JSON
json_folder = 'json_docs'

# Boucle pour lire chaque fichier JSON et l'indexer
for filename in os.listdir(json_folder):
    if filename.endswith('.json'):
        file_path = os.path.join(json_folder, filename)
        with open(file_path, 'r') as file:
            # Charger le contenu JSON
            document = json.load(file)
            # Ajouter le document à Solr
            solr.add([document])
        print(f"Indexed {filename}")
```

Conseils important sur l'indexation

- Toujours re-indexer les documents après une modification de configuration.

Exemple de requête et calcul des scores avec python

Requete et calcul des scores avec python

```
import pysolr

# Connexion à Solr
solr_url = 'http://localhost:8983/solr/BM25_sans_pret'
solr = pysolr.Solr(solr_url, always_commit=True)

# Terme de recherche
query = "Solr features"  # Remplacez par votre terme de recherche

# Exécuter la requête avec la récupération des scores
results = solr.search(query, **{
    'fl': '*,score',   # Inclut tous les champs et le score dans les résultats
    'rows': 10         # Nombre de résultats à récupérer (modifiable)
})

# Afficher les résultats avec leurs scores
for result in results:
    print(f"Document ID: {result['id']}")
    print(f"Title: {result.get('title', 'No Title')}")
    print(f"Score: {result['score']}")
    print("-" * 40)
```

Composants d'Analyse de Texte dans Lucene/PyLucene

AbstractWordsFileFilterFactory Classe parent abstraite pour les usines d'analyse qui acceptent un fichier de mots vides en entrée. Elle facilite la gestion et l'intégration des filtres personnalisés qui doivent exclure des mots spécifiques lors de l'indexation, améliorant ainsi la qualité de l'index en éliminant les termes non pertinents.

EnglishAnalyzer Analyseur spécialisé pour l'anglais. Il combine plusieurs filtres et techniques de traitement du texte, tels que la suppression des mots vides, le stemming, et la gestion des possessifs. Cela permet de normaliser les termes et d'améliorer la pertinence des recherches en anglais.

EnglishMinimalStemFilter TokenFilter qui applique un stemming minimal spécifique à l'anglais. Le stemming réduit les mots à leur racine, ce qui permet de regrouper différentes formes d'un même mot (par exemple, "running" et "runner" deviennent "run"). Cela améliore la recherche en permettant de retrouver toutes les variantes d'un mot racine.

EnglishMinimalStemFilterFactory Factory pour `EnglishMinimalStemFilter`. Elle facilite l'intégration du filtre de stemming minimal dans le pipeline d'analyse en fournissant une méthode standardisée pour son instanciation et sa configuration.

Composants d'Analyse de Texte dans Lucene/PyLucene (suite)

EnglishPossessiveFilter TokenFilter qui supprime les possessifs (comme le "'s") des mots en anglais. Par exemple, "John's" devient "John". Cela aide à normaliser les termes et à éviter la duplication des mots dus aux possessifs, améliorant ainsi la qualité de l'indexation.

EnglishPossessiveFilterFactory Factory pour `EnglishPossessiveFilter`. Elle permet d'intégrer facilement le filtre de suppression des possessifs dans le pipeline d'analyse, assurant une gestion cohérente et réutilisable des possessifs dans différents contextes d'indexation.

KStemFilter Filtre de stemming performant pour l'anglais, utilisant l'algorithme K-stem, optimisé pour être rapide tout en conservant une précision élevée dans la réduction des mots à leurs racines. Cela permet d'améliorer l'efficacité de l'indexation et de la recherche en anglais.

KStemFilterFactory Factory pour `KStemFilter`. Elle facilite l'intégration et la configuration du filtre K-stem dans le pipeline d'analyse, garantissant une application uniforme du stemming dans divers scénarios d'indexation.

Composants d'Analyse de Texte dans Lucene/PyLucene (suite)

PorterStemFilter TokenFilter qui transforme le flux de tokens selon l'algorithme de stemming de Porter. Le stemming de Porter est l'un des algorithmes de réduction des mots les plus utilisés, qui convertit les mots en leurs racines communes. Cela permet de regrouper des termes similaires, améliorant la pertinence et la performance des recherches textuelles.

PorterStemFilterFactory Factory pour `PorterStemFilter`. Elle agit en tant que factory pour créer des instances de `PorterStemFilter`, simplifiant l'intégration du filtre de stemming de Porter dans le pipeline d'analyse et assurant une application cohérente de l'algorithme de stemming dans le processus d'indexation des documents.

Exemple de recherche d'informations selon la similarity BM25

- Exemple de recherche d'informations selon la similarity BM25 sans prétraitement.

Exemple de recherche d'informations selon la similarity BM25 (suite)

- Exemple de recherche d'informations selon la similarity BM25 sans prétraitement (suite)

Merci pour votre attention !

UQÀM
Université du Québec à Montréal

Quelques références

[1] *Solr downloads*, `https://solr.apache.org/downloads.html`

[2] *Solr 9.7.0 documentation*, `https://cfd.cryptotradecorp.co/`

[3] *PySolr 3.10.0 documentation*, `https://pypi.org/project/pysolr/`

[4] *Solr definition, wikipedia*, `https://fr.wikipedia.org/wiki/Apache_Solr`