--- Commande curl pour envoyer un document de requetes:

```
curl -X POST -H "Content-Type: application/json"      -d
@requetes_courtes1_50.json
"http://localhost:8983/solr/Bm25_baseline/select?q=text:%22REQUETE%22%20OR
%20head:%22REQUETE%22&fl=id_doc,score&rows=1000&qf=head,%20text,%20SCHEMA"
-o results.txt
```

---- Commande curl pour indexer l'ensemble de tous les documents

##

Commande curl (terminal) pour indexer l'ensemble des documents existant dans un un dossier dans un core:

```
bin/post -c Bm25_baseline1 ./tous_lots_docs/*.json   (durée: 30 s)
```

##

Suppression de toutes les collections

```
# Remplacez par l'URL de votre Solr SOLR_URL="http://localhost:8983/solr" #
Étape 1 : Récupérer la liste des collections collections=$(curl -s "$
{SOLR_URL}/admin/collections?action=LIST" | jq -r '.collections[]') # Étape 2 :
Supprimer chaque collection for collection in $collections; do echo "Suppression
de la collection : $collection" curl -X GET "${SOLR_URL}/admin/collections?
action=DELETE&name=${collection}" -s | jq done echo "Toutes les collections ont
été supprimées."
```

##

```
import pysolr
import json

# Connexion à Solr (remplacez l'URL et le nom du core par les vôtres)
solr = pysolr.Solr('http://localhost:8983/solr/your_core_name', timeout=10)

# Chargement des requêtes depuis le fichier JSON
with open('requetes.json', 'r') as f:
    requetes = json.load(f)

# Nom du fichier de sortie pour TREC_Eval
output_file = 'results_trec_eval.txt'
run_id = "solr_run"  # Identifiant unique de l'exécution

# Ouverture du fichier de sortie pour enregistrer les résultats
with open(output_file, 'w') as outfile:
    # Boucle sur chaque requête dans le fichier JSON
    for query_id, query_text in requetes.items():
        # Exécute la requête dans Solr
        results = solr.search(query_text, rows=1000)  # Ajustez `rows` selon le
nombre de documents à récupérer

        # Traitement des résultats pour chaque document
        for rank, doc in enumerate(results, start=1):
            doc_id = doc['id']  # Assurez-vous que `id` correspond au champ ID
du document dans Solr
```

```
            score = doc['score']  # Assurez-vous que `score` est bien retourné
par Solr

            # Enregistre le résultat au format TREC_Eval
            outfile.write(f"{query_id} Q0 {doc_id} {rank} {score} {run_id}\n")

print(f"Les résultats ont été sauvegardés dans le fichier {output_file}")

################################################################################

### Liste des similarity dans solr

BM25SimilarityFactory, BooleanSimilarityFactory,
ClassicSimilarityFactory, DFISimilarityFactory,
DFRSimilarityFactory, IBSimilarityFactory,  LMDirichletSimilarityFactory,
LMJelinekMercerSimilarityFactory
:

(https://solr.apache.org/docs/9_7_0/core/org/apache/solr/schema/
SimilarityFactory.html)

Commande pour lister les fieldtypes :
curl http://localhost:8983/solr/<collection_name>/schema/fieldtypes



commande curl pour configurer un fieldtype :

-- BM25

(https://solr.apache.org/guide/6_6/field-types-included-with-solr.html)




NB : Dans Solr, si vous définissez un tokenizer et que vous n'ajoutez pas de
filtres, seul le tokenizer sera utilisé.

KeywordTokenizerFactory permet d’empêcher toute modification ou découpage.

Configuration sans pretraitement

Similarity Bm25 :

curl -X POST -H "Content-Type: application/json" \
http://localhost:8983/solr/<collection_name>/schema \
--data-binary '{
  "add-field-type":{
  "name": "long_string_base_bm25",
  "class": "solr.TextField",
  "similarity": {
    "class": "solr.BM25SimilarityFactory",
    "k1": 1.2,
    "b": 0.75
  },
  "analyzer": {
      "tokenizer": { "class": "solr.StandardTokenizerFactory" }
    }
}
}'
```

Similarity TFidf :

```
curl -X POST -H "Content-Type: application/json" \\
http://localhost:8983/solr/Tfidf_standard/schema \
--data-binary '{
  "add-field-type": {
    "name": "long_string_base_tfidf",
    "class": "solr.TextField",
    "similarity": {
      "class": "solr.ClassicSimilarityFactory",
      "discountOverlaps": true
    },
    "analyzer": {
      "tokenizer": { "class": "solr.StandardTokenizerFactory" }
    }
  }
}'
```

Small string :

--- Similarity Bm25 :

```
curl -X POST -H "Content-Type: application/json" \
http://localhost:8983/solr/<collection_name>/schema \
--data-binary '{
  "add-field-type": {
    "name": "small_string_bm25",
    "class": "solr.StrField",
    "similarity": {
      "class": "solr.BM25SimilarityFactory",
      "k1": 1.2,
      "b": 0.75
    }
  }
}'
```

Similarity TFidf :

```
curl -X POST -H "Content-Type: application/json" \
http://localhost:8983/solr/Tfidf_standard/schema \
--data-binary '{
  "add-field-type": {
    "name": "small_string_tfidf",
    "class": "solr.StrField",
    "similarity": {
      "class": "solr.ClassicSimilarityFactory",
      "discountOverlaps": true
    }
  }
}'
```

Configuration avec pretraitement

Difference entre filter et analyser

Aspect
Analyzer
Filter
Rôle
Organise la chaîne complète d’analyse (tokenizer + filtres).
Applique une transformation spécifique sur les tokens.
Composants
Contient au moins un tokenizer et peut contenir des filtres.
Transforme uniquement les tokens fournis par le tokenizer.
Obligation
Obligatoire dans un solr.TextField.
Optionnel : peut être ajouté ou non selon les besoins.
Position
Niveau global de la chaîne d’analyse.
Étape intermédiaire dans l’analyse.
Exemple
Définit un tokenizer et une série de filtres.
Convertit les tokens en minuscules ou supprime des mots vides.

```
"analyzer": {
  "tokenizer": { "class": "solr.StandardTokenizerFactory" },
  "filters": [
    { "class": "solr.LowerCaseFilterFactory" },
    { "class": "solr.StopFilterFactory", "words": "stopwords.txt", "ignoreCase":
true }
  ]
}
```

Étapes de transformation :

1. Tokenizer (StandardTokenizerFactory) :
    - Segmente le texte en tokens : ["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"].
2. Filter 1 (LowerCaseFilterFactory) :
    - Convertit chaque token en minuscules : ["the", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"].
3. Filter 2 (StopFilterFactory) :
    - Supprime les mots vides (the, over) : ["quick", "brown", "fox", "jumps", "lazy", "dog"].

Résultat final :

```
css
Copier le code
["quick", "brown", "fox", "jumps", "lazy", "dog"]

#####Suppression de fieldtype####################
curl -X POST -H "Content-Type: application/json" \
http://localhost:8983/solr/my_collection/schema \
--data-binary '{
  "delete-field-type": {
    "name": "long_string_base_tfidf"
  }
}'

#### Suppression d’une collection

curl -X GET "http://localhost:8983/solr/admin/collections?
action=DELETE&name=<collection_name>"



### Verification de la suppression d’une collections
```

```
curl "http://localhost:8983/solr/admin/collections?action=LIST"



Configuration des fieldtypes avec les similarities


--- BM25

curl -X POST -H "Content-Type: application/json" \
http://localhost:8983/solr/<collection_name>/schema \
--data-binary '{
  "add-field-type": {
    "name": "long_string_analyse_bm25",
    "class": "solr.TextField",
    "similarity": {
      "class": "solr.BM25SimilarityFactory",
      "k1": 1.2,
      "b": 0.75
    },
    "analyzer": {
      "tokenizer": { "class": "solr.StandardTokenizerFactory" },
      "filters": [
        { "class": "solr.LowerCaseFilterFactory" },
        { "class": "solr.StopFilterFactory", "ignoreCase": true, "words":
"stopwords.txt" },
        { "class": "solr.PorterStemFilterFactory" }
      ]
    }
  }
}'



--- Tfidf

curl -X POST -H "Content-Type: application/json" \
http://localhost:8983/solr/<collection_name>/schema \
--data-binary '{
  "add-field-type": {
    "name": "long_string_analyse_tfidf",
    "class": "solr.TextField",
    "similarity": {
      "class": "solr.ClassicSimilarityFactory",
      "discountOverlaps": true
    },
    "analyzer": {
      "tokenizer": { "class": "solr.StandardTokenizerFactory" },
      "filters": [
        { "class": "solr.LowerCaseFilterFactory" },
        { "class": "solr.StopFilterFactory", "ignoreCase": true, "words":
"stopwords.txt" },
        { "class": "solr.PorterStemFilterFactory" }
      ]
    }
  }
}'

### Ajout et configuration d’un champ ###

curl -X POST -H "Content-Type: application/json" \
http://localhost:8983/solr/<collection_name>/schema \
--data-binary '{
```

```
  "add-field": {
    "name": "<field_name>",
    "type": "<field_type>",
    "stored": true,
    "indexed": true
  }
}'
```