Forestier Quentin & Herzig Melvyn

MAC – 25.11.2021

Indexing and Search with Apache LuceneProtocoles applicatifs

Labo 3



Table des matières

[Familiarisation avec Lucene 2](#_Toc88687243)

[Indexing and Searching the CACM collection 4](#_Toc88687244)

[Indexing 4](#_Toc88687245)

# Familiarisation avec Lucene

**Quels sont les types des *fields* dans l’index :**

* StringField
* LongPoint
* TextField

**Quels sont les caractéristiques des types précédents :**

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Indexing | Storage | tokenization |
| StringField | Oui | Oui | Non |
| LongPoint | Sous forme de long | Non | Non |
| TextField | Oui | Non | Oui |

**Est-ce que la démo de la ligne de commande retire les stopwords :**

Non, le queryParser est fait avec un StandardAnalyzer et son constructeur vide qui n’utilise aucun stop words.



Pour les appliquer il aurait fallu les lui passer à la construction.

**Est-ce que la démo de la ligne de commande utilise un filtre de stemming :**

Non, le queryParser est fait avec un StandardAnalyzer qui n’utilise pas le filtre PorterStemFilter.

Comme le prouve notre test, apples n’est pas transformé en apple. De plus, ils ne retournent pas les même documents.

Une image contenant texte

Description générée automatiquement

**Est-ce que la démo de la ligne de commande est non sensible à la casse ?**

Oui car le StandardAnalyzer utilise un LowerCaseFilter.

Une image contenant texte

Description générée automatiquement

Comme nous pouvons le voir, les deux requêtes retournent les mêmes documents.

**Est-ce important de faire le stemming avant ou après le retrait des stopwords ?**

D’une manière générale, nous estimons qu’il est plus sage de retirer les stopwords avant de stemmer. Dans le cas inverse, nous n’excluons pas qu’un mot stemmé deviennent un stopword et soit retiré de la requête.

# Indexing and Searching the CACM collection

## Indexing

**Qu’est qu’un vecteur de termes ?**

Selon [northcoder](https://northcoder.com/post/lucene-fields-and-term-vectors/#:~:text=Term%20vectors%20are%20an%20alternative,for%20it%20to%20be%20created.&text=Note%20that%20you%20need%20to,term%20vector%20positions%20and%20offsets.), un vecteur de terme est une manière alternative de structurer des données indexables. Par défaut, un vecteur de terme n’est pas stocké dans l’index.

Les données sont normalement indexées de la manière suivante :  
field > term > doc > freq/pos

À l’inverse, le vecteur :  
doc > field > term > freq/pos

Nous constatons que l’indexation n’est plus faite selon les champs mais selon les documents

**Que faut-il ajouter dans le code pour accéder au vecteur de termes ?**

La méthode la plus basique est :

summaryField.setStoreTermVectors(true);

Mais il existe également :

summaryField.setStoreTermVectorOffsets(true);

summaryField.setStoreTermVectorPayloads(true);

summaryField.setStoreTermVectorPositions(true);

Après avoir ajouté la première méthode, si un document présente le champ *summary* nous pouvons accéder à son vecteur de terme avec un clic droit.



Ce qui ouvre le vecteur de terme du document en question.

Une image contenant table

Description générée automatiquement

**Comparer la taille de l’index avant et après avoir activé les vecteurs de termes. Discuter du résultat.**

|  |  |
| --- | --- |
| Sans vecteur de termes | Avec vecteur de terme |
|  |  |

L’index avec vecteur de terme basique est environ 1,5 fois plus lourd que l’index sans vecteur de terme. Pourquoi ? L’index sans vecteur de termes stocke uniquement des informations générales pour chercher un document. Cependant, l’index avec vecteur de terme stocke, en plus, des information de recherche spécifique à chaque document. Nous comprenons alors que ce second index stocke une structure de données supplémentaire par document, ce qui alourdit l’index.

Les indexes sans vecteur de termes sont utilisés pour chercher un document parmi une collection. Une fois le document trouvé, si nous bénéficions d’un vecteur de terme, nous pouvons affiner la recherche en récupérant où est située l’information pertinente.

**Code ajouté dans CACMIndexer.onNewDocument :**

doc.add( new StoredField("id", id));

for(String author: authors.split(";")) {

  doc.add( new StringField("authors", author, Field.Store.YES));

}

doc.add( new StringField("title", title, Field.Store.YES));

if(summary != null) {

  FieldType summaryField = new FieldType();

  summaryField.setIndexOptions(IndexOptions.DOCS\_AND\_FREQS\_AND  
 \_POSITIONS\_AND\_OFFSETS);

  doc.add(new Field("summary", summary, summaryField));

}

## Utiliser différents analyseurs