Overvier

Khi một văn bản được index, các field của nó là các đối tượng để analyzing và tokening để chuyển đổi và bình thường hóa dữ liệu các field đó. Vd: bỏ khoảng trắng, bõ tag html, stemming, bỏ các ký tự đặc biệt và thay bằng các ký tự khác…

1. Stemming

Có 3 loại stemming

* Porter stemming (Reduction stemming): Thuật toán chuyển đổi nhằm giảm thiểu các hình thức của 1 từ về từ gốc (1 từ chia ở các thì khác nhau trong tiếng anh: runs, ran, running về từ gốc là run). Được sử dụng trong cả quá trình insert và query.
* Expansion stemming: Ngược lại với Reduction stemming. Thuật toán xuất phát từ 1 từ gốc rồi phát sinh (expand) ra các hình thức của từ đó. Được sử dụng trong insert và query. (Sử dụng SynonymFilterFactory)
* Kstem: một sự thay thế cho Porter stemming.

1. Analyzer

* Là thành phần tiền xử lý các dữ liệu text khi index và search.

#### 3. [CharFilterFactories](http://wiki.apache.org/solr/AnalyzersTokenizersTokenFilters#CharFilterFactories)

#### 3.1 solr.PatternReplaceCharFilterFactory

* **PatternReplaceFilterFactory**: Takes a regular expression and replaces the matches. Example:

<filter class="solr.PatternReplaceFilterFactory" pattern=".\*@(.\*)" replacement="$1" replace="first" />

1. **TokenFilterFactories**

#### solr.StandardFilterFactory

org.apache.lucene.analysis.standard.StandardFilter

Loại bỏ dấu ‘.’ trong các từ viết tắt và ‘s khi kết thúc 1 token. Chỉ hoạt động trên “typed token” – được cung cấp bởi StandardTokenizer hoặc tương đương.

Vd: StandardTokenizer với StandardFilter

"I.B.M. cat's can't" ==> "IBM", "cat", "can't"

#### solr.LowerCaseFilterFactory

org.apache.lucene.analysis.LowerCaseFilter

Chuyển các ký tự trong token về chữ thường

"I.B.M.", "Solr" ==> "i.b.m.", "solr".

#### solr.TrimFilterFactory

org.apache.solr.analysis.TrimFilter

Xóa bỏ khoảng trắng ở đầu và cuối các token.

Vd: " Kittens!   ", "Duck" ==> "Kittens!", "Duck".

updateOffsets: thuộc tính cập nhật lại vị trí offset bắt đầu và kết thúc

#### solr.StopFilterFactory

org.apache.lucene.analysis.StopFilter

Loại bỏ các stop word.

Các stop word mặc định

"a", "an", "and", "are", "as", "at", "be", "but", "by",

"for", "if", "in", "into", "is", "it",

"no", "not", "of", "on", "or", "s", "such",

"t", "that", "the", "their", "then", "there", "these",

"they", "this", "to", "was", "will", "with"

Có thể tùy chỉnh các stop word với thuộc tính “words” trong file schema. Thuộc tính ”inogeCase” được sử dụng để bỏ qua việc so sánh của 1 thẻ với danh sách stop word.

Vd:

<fieldtype name="teststop" class="solr.TextField">

<analyzer>

<tokenizer class="solr.LowerCaseTokenizerFactory"/>

<filter class="solr.StopFilterFactory" words="stopwords.txt" ignoreCase="true"/>

</analyzer>

</fieldtype>

#### solr.CommonGramsFilterFactory

org.apache.solr.analysis.CommonGramsFilter

Dùng để phát sinh các câu truy vấn có chứa cả các stop word. Vd: Bình thường khi tìm kiếm từ khóa “the cat”, thì stop word sẽ bị loại bỏ, khi đó việc tìm kiếm sẽ search trên các document có chứa từ “cat”. Khi index thì các gram phổ biến được index dưới dạng như “the cat” 🡪 “the\_cat”. CommonGramsQueryFilter chuyển đổi các cụm từ tìm kiếm như “the cat” thành 1 term query “the\_cat”.

Danh sách các common word được xác định trong thuộc tính “words” của schema. Thuộc tính “inorgeCase” dùng để bỏ qua các trường hợp khi so sánh với danh sách common word.

#### solr.KeepWordFilterFactory

org.apache.solr.analysis.KeepWordFilter

Ngược lại với StopFilterFactory. Giữ lại các từ trong danh sách.

<fieldtype name="testkeep" class="solr.TextField">

<analyzer>

<filter class="solr.KeepWordFilterFactory" words="keepwords.txt" ignoreCase="true"/>

</analyzer>

</fieldtype>

#### solr.LengthFilterFactory

solr.LengthFilter

Lọc ra những token có dạng \*not\* có chiều dài min 🡪 max

<fieldtype name="lengthfilt" class="solr.TextField">

<analyzer>

<tokenizer class="solr.WhitespaceTokenizerFactory"/>

<filter class="solr.LengthFilterFactory" min="2" max="5" />

</analyzer>

</fieldtype>

#### solr.PorterStemFilterFactory

 org.apache.lucene.analysis.PorterStemFilter

Xuất phát từ thuật toán Porter Stemming, loại bỏ các kết thúc từ.

Vd:  "riding", "rides", "horses" ==> "ride", "ride", "hors"

#### solr.EnglishPorterFilterFactory

#### solr.SnowballPorterFilterFactory

#### solr.WordDelimiterFilterFactory

solr.analysis.WordDelimiterFilter

Chia nhỏ các word thành các subword theo nguyên tắc sau.

* Chia bởi các dấu phân cách (Không phải các ký tự alpha và chữ số)

"Wi-Fi" -> "Wi", "Fi"

* Chia trong trường hợp chuyển tiếp

"PowerShot" -> "Power", "Shot"

* Chia trong trường hợp chuyển tiếp từ chữ cái sang số

"SD500" -> "SD", "500"

* Các từ phân cách ở đầu hay cuối mỗi subword đều loại bỏ

"//hello---there, 'dude'" -> "hello", "there", "dude"

* Loại bỏ chữ ‘s’ ở các subword

"O'Neil's" -> "O", "Neil"

Việc chia nhỏ các subword bị ảnh hưởng bởi các tham số

**splitOnCaseChange="1"**

**splitOnNumerics="1"**

**stemEnglishPossessive="1" (loại bỏ chữ s)**

**generateWordParts="1"**

* + "PowerShot" => "Power" "Shot" (if splitOnCaseChange=1)
  + "Power-Shot" => "Power" "Shot"

**generateNumberParts="1"** tạo ra các subword dạng number

* + "500-42" => "500" "42"

**catenateWords="1"** kết hợp các subword dạng chữ cái 🡪 1 chuỗi

* + "wi-fi" => "wifi"

**catenateNumbers="1"** : kêt hợp tất cả các subword dạng number 🡪 1 chuỗi

* + "500-42" => "50042"

**catenateAll="1"** kết hợp tất cả subword thành 1 chuỗi

* + "wi-fi-4000" => "wifi4000"

**preserveOriginal="1"** giữ nguyên token để index (không sửa đổi).

Có thể kết hợp các tham số lại với nhau.

Vd:

* generateWordParts="1" and catenateWords="1":
  + "PowerShot" 🡪 0:"Power", 1:"Shot" 1:"PowerShot"   
    (0,1,1 là vị trí các token)
  + "A's+B's&C's" -> 0:"A", 1:"B", 2:"C", 2:"ABC"
  + "Super-Duper-XL500-42-AutoCoder!" 🡪 0:"Super", 1:"Duper", 2:"XL", 2:"SuperDuperXL", 3:"500" 4:"42", 5:"Auto", 6:"Coder", 6:"AutoCoder"

\*\*Sử dụng trong quá trình index và query

<fieldtype name="subword" class="solr.TextField">

<analyzer type="query">

<tokenizer class="solr.WhitespaceTokenizerFactory"/>

<filter class="solr.WordDelimiterFilterFactory"

generateWordParts="1"

generateNumberParts="1"

catenateWords="0"

catenateNumbers="0"

catenateAll="0"

preserveOriginal="1"

/>

<filter class="solr.LowerCaseFilterFactory"/>

<filter class="solr.StopFilterFactory"/>

<filter class="solr.EnglishPorterFilterFactory"/>

</analyzer>

<analyzer type="index">

<tokenizer class="solr.WhitespaceTokenizerFactory"/>

<filter class="solr.WordDelimiterFilterFactory"

generateWordParts="1"

generateNumberParts="1"

catenateWords="1"

catenateNumbers="1"

catenateAll="0"

preserveOriginal="1"

/>

<filter class="solr.LowerCaseFilterFactory"/>

<filter class="solr.StopFilterFactory"/>

<filter class="solr.EnglishPorterFilterFactory"/>

</analyzer>

</fieldtype>

#### solr.SynonymFilterFactory

SynonymFilter.

Tìm kiếm chuỗi các token và thay thế nó bằng các chuỗi token khác. (Cùng nghĩa)

**Synonyms:** tham sốxác định file định nghĩa các synonym (từ đồng nghĩa)

Nếu **inorgeCase = true:** tìm kiếm chữ thường trước khi tìm kiếm tương đồng

Nếu **expand = true:** việc tìm kiếm sẽ được mở rộng bằng cách thay thế tất cả các từ đồng nghĩa (synonym). Nếu bằng **false** tất cả sẽ được thay thế bằng từ đầu tiên trong danh sách.

<fieldtype name="syn" class="solr.TextField">

<analyzer>

<tokenizer class="solr.WhitespaceTokenizerFactory"/>

<filter class="solr.SynonymFilterFactory synonyms="syn.txt" ignoreCase="true" expand="false"/>

</analyzer>

</fieldtype>

Định dạng file syn.txt

# blank lines and lines starting with pound are comments.

#Explicit mappings match any token sequence on the LHS of "=>"

#and replace with all alternatives on the RHS. These types of mappings

#ignore the expand parameter in the schema.

#Examples:

i-pod, i pod => ipod,

sea biscuit, sea biscit => seabiscuit

#Equivalent synonyms may be separated with commas and give

#no explicit mapping. In this case the mapping behavior will

#be taken from the expand parameter in the schema. This allows

#the same synonym file to be used in different synonym handling strategies.

#Examples:

ipod, i-pod, i pod

foozball , foosball

universe , cosmos

# If expand==true, "ipod, i-pod, i pod" is equivalent to the explicit mapping:

ipod, i-pod, i pod => ipod, i-pod, i pod

# If expand==false, "ipod, i-pod, i pod" is equivalent to the explicit mapping:

ipod, i-pod, i pod => ipod

#multiple synonym mapping entries are merged.

foo => foo bar

foo => baz

#is equivalent to

foo => foo bar, baz

#### solr.RemoveDuplicatesTokenFilterFactory

org.apache.solr.analysis.RemoveDuplicatesTokenFilter.

#### solr.ISOLatin1AccentFilterFactory

org.apache.lucene.analysis.ISOLatin1AccentFilter

Bỏ dấu các ký tự Latin (ISO-8859-1)

#### solr.ASCIIFoldingFilterFactory

 org.apache.lucene.analysis.ASCIIFoldingFilter

Chuyển các ký tự unicode có mã lớn hơn 127 🡪 ký tự ansi nếu tồn tại.

#### solr.ShingleFilterFactory

Khởi tạo  shingles (token n-grams)  từ một token stream.

|  |  |
| --- | --- |
| **arg** | **value** |
| maxShingleSize | default 2 |
| outputUnigrams | default true |

<filter class="solr.ShingleFilterFactory" maxShingleSize="2" outputUnigrams="true"/>

Vd:

please divide this sentence into shingles"

🡪 "please divide", "divide this", "this sentence", "sentence into", "into shingles".

#### solr.PositionFilterFactory