**LUCENE IN ACTION**

# CHAPTER 4: ANALYSIS

Trong chương này những vấn đề được đặt ra và giải quyết:

* Tìm hiểu quá trình phân tích của Lucene.
* Khảo sát vấn đề quan trọng đó là QueryParse.
* Khảo sát quá trình dùng các hàm của Lucene để tạo bộ phân tích theo mục đích riêng.
* Vấn đề tích hợp các ngôn ngữ trong Lucene.

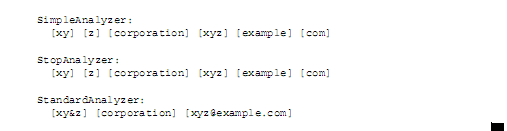
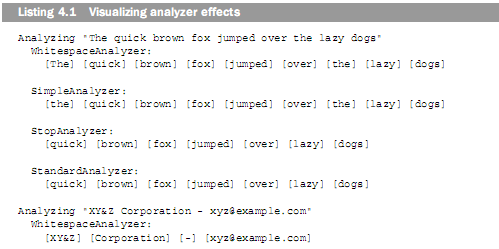
Trong Lucene , phân tích (*analysis*) là quá trình chuyển đổi các field văn bản về dạng trình bày chỉ mục cơ bản nhất (*term*).Các terms thì được sử dụng để xác định rõ tài liệu nào sẽ phù hợp với một câu truy vấn trong quá trình tìm kiếm.Bộ phân tích (*analyzer*) là cách nói tóm lược quá trình phân tích . Analyzer phân tích trong đoạn văn bản thành *tokenizes* , đó là quá trình rút trích các từ, bỏ đi hệ thống các dấu chấm câu , chuyển toàn bộ các chữ trong văn bản về dạng chữ thường ( *lowercasing* hay còn gọi là *normalizing*), loại bỏ các từ chung (*common words*) , giảm số lượng từ từ văn bản đưa vào (*root form* hay còn gọi *stremming*). Quá trình này còn được gọi là *tokenization,*chuyển đoạn văn bản thành nhiều khúc văn bản được gọi là các *token*, (sẽ được nói rõ hơn trong các phần sau) *.*Tokens được kết hợp với các field name của chúng được gọi là *terms* .

Mục đích chính của Lucene là kết quả có thể thu hồi được thông tin dễ dàng , vì vậy điều quan trọng đó là quá trình làm tinh dữ liệu từ dữ liệu thô đưa vào.Sau quá trình tạo ra terms , terms sẽ là những khối dữ liệu được dùng để tìm kiếm trực tiếp. Vì vậy chọn bộ phân tích đúng đắn là cốt yếu quan trọng của quá trình phát triển phần mềm tìm kiếm. Ngôn ngữ là một yếu tố phải được nghĩ đến để chọn bộ phân tích, bởi vì đều có đặc trưng riêng và duy nhất của từng ngôn ngữ. Kế tiếp đó là phạm vi của đoạn text , ví dụ tìm kiếm văn bản sẽ có phạm vi tìm kiếm khác với tìm kiếm media. Phải quan tâm đến các yếu tố khác như các thuật ngữ, thành ngữ, các từ viết tắt từ các bộ chữ ghép lại (ví dụ như NATO:North Atlantic Treaty Organization), … Lucene là nền tảng để chúng ta có thể tạo ra bộ tìm kiếm theo ý muốn .

Using analyzers

Quá trình phân tích diễn ra tại 2 nơi : trong suốt quá trình indexing và khi sử dụng QueryParser.

Ví dụ một quá trình phân tích nhóm từ: “The quick brown fox jumped over the lazy dogs” và “XY&Z Comporation – xyz@example.com”.Với từng token được đưa ra giữa các dấu [] để tạo sự phân chia rõ ràng. Điều quan trọng nhất là các terms indexed thì là các terms có thể dùng để tìm kiếm.



Lucene không tạo ra kết quả phân tích có thể nhận thấy được đối với khách hàng (*end user*). Terms được tách ra từ đoạn văn bản nguyên bản để indexing và tìm phù hợp trong suốt trong quá trình searching. Khi searching với QueryParser quá trình phân tích diễn ra trở lại để đảm bảo kết quả phù hợp tốt nhất.

### Indexing analysis

Trong suốt quá trình index , một Analyzer được nắm giữ từ IndexWriter.Với từng tokenized field của từng tài liệu được index sử dụng *IndexWriter* được sử dụng analyzer tùy trường hợp.Có hai loại Field đặc biệt được sử dụng để tạo tokennized là Text và Unstored. Khi index, có thể lựa chọn analyzer lúc IndexWriter hoặc theo từng cấp per-Document.

Field.KeyWord indexed field không phải là tokenized.Nó là một terms đơn lẻ với các giá trị đúng như một bộ cung cấp

### QueryParser analysis

Analyzer là chìa khóa để terms indexed. Vì vậy phải cần chắc chắn rằng khi truy vấn phải đưa được chính xác terms indexed hợp trong tài liệu tìm kiếm hợp lệ. Khi sử dụng hàm API-created truy vấn như là TermQuery, người phát triển phần mềm cần phải chắc chắn đảm bảo rằng terms được sử dụng phải tương ứng khi indexed. QueryParser sử dụng analyzer để có thể làm tốt nhất công việc tìm terms tương ứng khi indexed. QueryParser phân tích chia các mẫu của biểu thức khi sử dụng câu lệnh:

Query query = QueryParser.parse(expression,”contents”,analyzer);

Biểu thức bao gồm các toán tử, các loại dấu ngoặc đơn, kép , các biểu thức cú pháp đặc biệt khác để biểu thị xếp loại , fuzzy searches.

### Parsing versus analysis : khi một analyzer không phù hợp

Điều quan trọng của analyzers là khi sử dụng bên trong cho việc đánh dấu thành tokenized. Các tài liệu kiểu HTML, Microsoft Word, XML,… chứa các meta-dat như author, title, last modified date, potentially,… Khi indexing các tài liệu như kiểu meta-data có thể bị tách rời và indexed tạo thành fileds tách rời nhau.

Analyzers được sử dụng để phân tích không thể giúp được gì cho field separation vì các dãy thì được thỏa thuận với các field đơn tại một thời điểm.Trong những trường hợp này dữ liệu cần được parsed hoặc tiền xử lý , chuyển các separate blocks tách rời của đoạn text thành từng field (Chapter 7 sẽ nói rõ phần này).

## Analyzing the analyzer

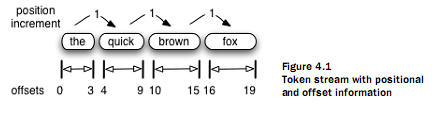
Điều quan trọng để xây dựng một bộ analyzer là phải biết rõ kiến trúc và xây dựng blocks provied, sử dụng Lucene đánh giá đúng mức và phải hiểu rõ bộ phân tích của Lucene, phải biết mở rộng và sửa lại một vài chỗ chưa hoàn hảo.

Analyzer là lớp cơ bản.Trong phần này sẽ nói rõ về từng thành phần chính được sử dụng bởi analyzers, bao gồm cả Token và TokenStream .

### What’s in a token?

Một stream của tokens là đầu ra cơ bản của quá trình phân tích. Trong suốt quá trình indexing , fields được chỉ định cho tokenization thì được xử lý với analyzer, với mỗi token thì được viết ra trở thành index như là một term.

Ví dụ , khi analyzer đoạn text “the quick browm fox”. Với từng token trình bày các từ riêng biệt của đoạn text. Một token mang theo một giá trị của đoạn text và một vài giá trị sau khi dữ liệu được phân tích đó là offsets của điểm bắt đầu và kết thúc trong đoạn text ban đầu, loại token, sự độ tăng về vị trí (*position increment)* .



Start Offset là vị trí của ký tự đầu tiên trong đoạn text ban đầu được xuất hiện trong vị trí bắt đầu của token, end offset là vị trí của ký tự cuối cùng của token text. Loại token có thể là một “ *String”* , mặc định là “*Word*”, có thể điều chỉnh lại được quá trình lọc token nếu muốn. Trong đoạn text được tokenize , vị trí liên qua đến token trước đó thì được ghi lại như giá trị *position increment.*

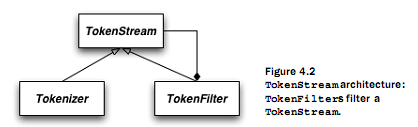
**Tokens into terms :** là đoạn text sau khi được analyzer trong suốt quá trình indexing, từng token được đặt đến index như một term. Position increment thì chỉ được thêm kết hợp meta-datavowis token được mang theo để index. Star offset và end offset là loại token thì bị loại bỏ , chúng chỉ được sử dụng trong quá trình phân tích.

**Position increments:** Là giá trị ghi lại sự tăng lên về vị trí của token hiện tại với token kế trước nó. Thông thường giá trị của nó là 1, chỉ ra với từng từ thì chỉ được chứ duy nhất và vị trí kế tiếp trong field. Vị trí này có giá trị lớn hơn 1 khi có một kẽ hở , từ tại vị trí đó trong quá trình index đã bị loại bỏ, ví dụ từ đó nằm trong tập hợp stop-word. Một token với một giá trị position increments là 0 khi nó có cùng vị trí như token trước nó. Analyzer xen vào một từ bí danh (*word aliases*).

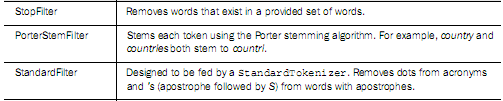
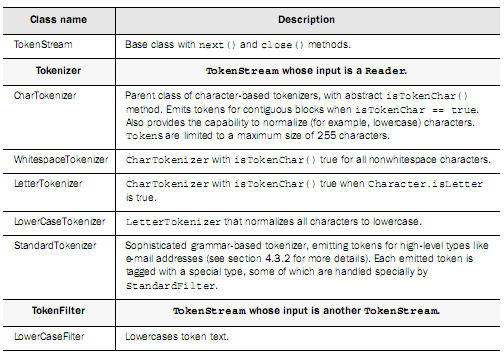
### TokenStreams uncensored

Có 2 loại TokenStream khác nhau : Tokenizer và TokenFilter. Tokenizer là liên kết của các ký tự riêng lẽ (*characters*), TokenFilter là liên kết của các từ (*words*).

Tokenizer là một TokenStream mà có tokenizes đưa vào từ một Reader, khi indexing một String trong Field.Text(String, String) hoặc Field.UnStore(String, String). Còn với loại TokenFilter thì được kết nối các TokenStream con lại . với từng TokenStream thì được xử lý kiểm tra xem là Tokenizer hay là TokenFilter để xử lý tiếp. TokenStream được thiết kế như mẫu Composite trong Design Pattern.



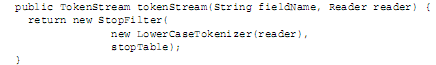
Các hàm bên trong:



Minh họa tổng quát, ở đây trung tâm là StopAnalyzer.

### 

Ví dụ một đoạn code trong StopAnalyzer:



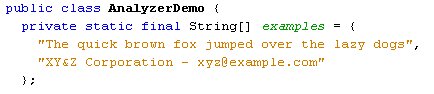
Trong StopAnalyzer này thì , LowerCaseTokenizer là đầu vào cung cấp cho StopFilter. LowerCaseTokenizer cho ra các token từ đoạn text ban đầu. Các lý tự không phải là chữ cái hoặc các chữ cái nằm trong Stop-Word list trở thành đường biên để tạo token và được loại ra ngoài.

Buffering là đặc trưng chung cần thiết trong TokenStream để nó có thể thực thi được. Low-Level tokenizers để buffer chứa characters từ form tokens khi chia được sự ngăn cách tại các vùng biên dựa vào whitespace, nonletter characters.

### Visualizing analyzers

Điều quan trọng để hiểu được sự khác nhau của các loại analyzer với dữ liệu đưa vào. Có 4 bộ analyzers , có thể thấy được sự khác nhau khi sử dụng AnalyzerDemo , trong hàm này đưa vào 2 đoạn text để kiểm tra 4 bộ phân tích. Đầu ra là kết quả sau khi được phân tích và các term được trình bày tách biệt thông qua dấu [] , tức là kết quả sau khi được indexing.

Hai đoạn text được đưa vào phân tích:

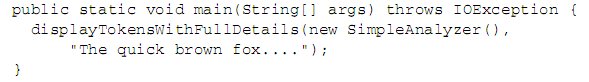


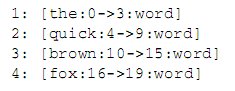
Kết quả:



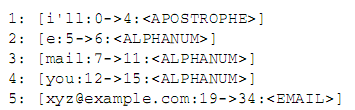
Một vài đặc điểm quan trọng của 4 bộ phân tích có trong Lucene:

* WhitespaceAnalyzer không thực hiện lowercase(chuyển tất cả sang ký tự thường ) , vẫn giữ dấu “-“ , loại bỏ tất cả các khoảng trắng, và dựa vào các khoảng trắng để làm đường biên phân chia tokenize.
* SimpleAnalyzer thực hiện lowercase, vẫn giữ các từ nằm trong danh sách stop-word, dùng các ký tự không phải là các chữ cái alphabetic để làm đường biên phân chia tokenize.
* Cả SimpleAnalyzer và StopAnalyzer đều làm hỏng tên của tên các công ty như ví dụ trên (xy&z bị rã ra thành [xy] [z] , loại bỏ ký hiêu &) .
* StopAnalyzer và StandarAnalyzer loại bỏ các từ nằm trong Stop-Word ví dụ như “*the*” ở ví dụ trên.
* StandarAnalyzer vẫn giữ được tên của công ty và thực hiện chức năng lowercase , loại bỏ dấu “-” , giữ được địa chỉ email (*xyz@example.com*).

**Looking inside tokens** TokenStream có thể tạo ra các token , TokenFilters có thể truy cập dữ meta-data . Thông qua ví dụ sau để giải thích cho việc truy cập dữ liệu data-meta :và kết quả đưa ra:

****

**What good are start and end offsets?**  Cả hai giá trị này không được sử dụng trong Lucene. Trong phần thảo luận section 8.7 về term highlighter sử dụng một TokenStream và kết quả là Tokens đưa ra ngoài để indexing xác định nơi nào a block của text bắt đầu và kết thúc highlighting, và đưa ra những từ trong kết quả tìm kiếm được cho là nổi bật nhất.

**Token-type usefulness** Hiển thị token-type để tạo sự riêng biệt trong từng loại của tokens. Ví dụ : **** kết quả đưa ra: 

### Filtering order can be important:

Phần này chủ yếu sử dụng ví dụ stop-word để loại bỏ các từ, ký tự không quan trọng trong dữ liệu đưa vào, tạo ra một bộ lọc làm tinh dữ liệu.

## Using the built-in analyzers

Lucene gồm có một vài bộ phân tích chính như : WhiteSpaceAnalyzer, SimpleAnalyzer, StopAnalyzer , standardAnalyzer và có hỗ trợ 2 ngôn ngữ đặc biệt : RussianAnalyzer, GermanAnalyzer. Tìm hiểu rõ StopAnalyzer và StandardAnalyzer trong phần này.

### StopAnalyzer

StopAnalyzer chia các từ cơ bản trong dữ liệu đưa vào và lowercasing , còn removes những stop words. Đưa vào các từ English thuộc stopword vào trong StopAnalyzer , ví dụ như list các từ : 

Bộ phân tích sau khi nhận dữ liệu đưa vào, rút gọn các từ chung được sử dụng trong English, ví dụ như *don’t , can’t, it’s.* Trước khi loại bỏ các từ có trong stop-word, StopAnalyzer giữ các ký tự lần lượt với nhau, ngăn chia tại các ký tự không phải là chữ cái bao gồm cả dấu “ ’ ” ( ví dụ như I’ll ).loại bỏ các ký tự như “s”, “t” đứng riêng lẻ, nó có khả năng phán đoán để loại bỏ các từ vô nghĩa.

Stop-word khi loại bỏ nảy sinh một số vấn đề như : làm gì với những lỗ hổng sau khi một số từ được loại bỏ ? Giả định bạn index “one is not enough”.Các token được tạo ra từ StopAnalyzer là [one] [enough] với “is” và “not” thì bị loại bỏ.Vì vậy kết quả sẽ đưa ra chính xác nếu như indexed có chứa dữ liệu “one enough”.Nếu như khi dữ dụng Query-Parser với StopAnalyzer, câu này sẽ dduwwocj match với các nhóm từ như “one enough”,”one is enough”,”one but not enought”, ”one is not enought”. Khi sử dụng QueryParser để phân tích câu trên, khi câu được chuyển sang các trường hợp khác nhau và khi đưa về trường hợp “one enough” thì được match với terms indexde. Đây chính là một “hole” trong topic này. Điều này sẽ được thảo luận kỹ trong phần 4.7.3.

Khi loại bỏ các từ thuộc list stop-word thì có thể ngữ nghĩa trong câu sẽ bị biến đổi, khả năng này có thể bị xảy ra nhưng cũng tùy vào ngữ cảnh mà nó mà tầm ảnh hưởng sẽ nhỏ lại, phụ thuộc vào việc sử dụng Lucene trong hoàn cảnh nào để tìm kiếm ý nghĩa của từ trong chương trình.

### StandardAnalyzwer

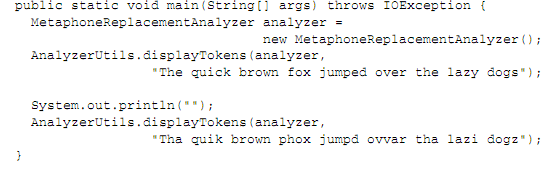
StandardAnalyzer là bộ phân tích chung nhất , nền tảng gắn liền với analyzer trong Lucene. Tokenizing thông minh với các loại từ vựng, chữ cái , viết tắt của các chữ đầu dòng, tên công ty, địa chỉ email, tên miền , số, serial number. IP address, CJK (Chinese, Japanese, Korean) character. Loại bỏ các từ nằm trong stopword , vì vậy StandardAnalyzer thường là lựa chọn đầu tiên.

## Dealing with keyword fields

Dễ dàng index một keyword bằng cách sử dụng Field.Keyword.Trong tìm kiếm media phần này không sử dụng, chủ yếu dùng cho tìm kiếm tin tức.

## “Sounds like” querying

Xử lý những từ đồng âm, khi người dùng tìm kiếm không ghi đúng từ cần tìm mà ghi lại từ đồng âm để tìm kiếm . Ví dụ như chuỗi ký tự đưa vào để tìm kiếm “Kool kat” tức là họ muốn tìm “cool cat”. Lucene cũng hỗ trợ để lập trình viên giải quyết được trường hợp này.



Kết quả đưa ra dữ liệu theo kiểu Metaphone Encoder :



“sounds-like” được thiết kế đặc trưng cho search engines cho trẻ em rất hữu dụng, hầu hết những sounds-like được thấy đều là những từ được viết sai chính tả , vì vậy nó cần thiết cho việc tra cứu hỏi ý kiến người sử dụng hiệu chỉnh lại từ khóa tìm kiếm khi cần thiết.

## Synonyms, aliases, and words that mean the same

## 

Analyzer thêm vào các từ đồng nghĩa trong chuỗi token được đưa ra ngoài , nhưng các từ được thay thế thì được đặt ở cùng vị trí trong từ gốc.

## Stemming analysis

Những lỗ hổng được dời đi ở những nơi stop words được loại bỏ khi thực hiện chức năng phân tích dùng PositionalStopFiler.

Những từ có dạng khác biệt của một từ thì được chuyển về dạng chung nhất, thường hay gặp trong tiếng Anh ví dụ như : breathe, breathes, breathing, breathed thông qua Porter stemmer thì đều được chuyển lại thành breath.

## Language analysis issues

Việc liên kết các ngôn ngữ khác nhau trong Lucene là một điều thú vị và có rất nhiều vấn đề phức tạp. Làm thế nào một đoạn text với các ngôn ngữ khác nhau được index và sau đó được tìm lại được? vì vậy sẽ có rất nhiều vấn đề mà người phát triển phần mềm cần xem xét.

Rào cản đầu tiên đó là các ký tự được mã hóa (encoding) , trong suốt quá trình phân tích các ngôn ngữ khác nhau thì có tập hợp stop-words khác nhau và có một thuật toán stremming algorithms duy nhất. Lucene có hỗ trợ một số bộ phân tích được xây dựng hỗ trợ cho developer như Tokenizers và TokenStreams có trong SandBox.

Vấn đề ngôn ngữ cần được xem xét trong luận văn đó là ngôn ngữ Unicode và encodings. Lucene có chứa tất cả các ký tự trong UTF-8 encoding chuẩn, đó chính là điều kiện thuận lợi để đưa vào dữ liệu encoding cho bộ phân tích. Khi đọc một tập tin HTML hoặc XML từ HTTP server , encoding chính là vấn đề gặp phải, các ký tự phức tạp gây khó khăn cho quá trình phân tích.

Ngoài ngôn ngữ English , Lucene còn hỗ trợ rất nhiều ngôn ngữ khác cho bộ phân tích .Ví dụ như GermanAnalyzer và RussianAnalyzer , ngoài ra còn có bộ SnowballAnalyzer hỗ trợ cho hầu hết các ngôn ngữ European . Bộ phân tích cho các ngôn ngữ tượng hình điển hình như các nước : Trung Quốc, Hàn Quốc, Nhật Bản (CJK ) . Chỉ có duy nhất StandardAnalyzer đi kèm trong bộ Analyzer là hữu dụng cho tất cả các ngôn ngữ châu Á. Tuy nhiên , có 2 bộ Analyzer trong Lucene Sandbox phù hợp cho các ngôn ngữ châu Á, húng thì không được đính kèm trong core Lucene .

Một vấn đề đặc biệt nữa cần quan tâm đó là dữ liệu đưa vào gồm nhiều ngôn ngữ.Khi bạn indexing tài liệu chứa nhiều ngôn ngữ từ một single index, sử dụng per-Document analyzer là thích hợp, tức là dung từng bộ phân tích cho từng tài liệu cho từng ngôn ngữ. Nếu vẫn phải bắt buộc thêm field dữ liệu đa ngôn ngữ, thì field đó cần sử dụng bộ lọc cho kết quả tìm kiếm , hoặc hiển thị mục đích trong suốt quá trình khôi phục

Bộ phân tích là một khía cạnh trong Lucene có nhiều vấn đề cần được quan tâm và nỗ lực giải quyết một cách xứng đáng.Sử dụng StandartAnalyzer là bí quyết để dùng cho quá trình indexing analysis cho nhiều ứng dụng chung , nhưng điều quan trọng trong quá trình phân tích , người sử dụng cung cấp một đoạn text tìm kiếm và thường có thể sẽ gây ra những kết quả nhầm lẫn , ví dụ khi tìm kiếm “to be or not to be” sẽ không cho một kết quả nào vì trong lúc quá trình analyzer các từ nằm trong bộ lọc đã loại bỏ các từ nằm trong danh sách StopWord.

Khi thay đổi bộ phân tích thì điều bắt buộc phải rebuild lại index và cũng sử dụng bộ phân trích mới cho tất cả tài liệu đã được phân tích trước đó.