Excel で学ぶ！Lucene のスコア計算

2011 年 3 月 24 日
株式会社 ロンウイット　　阿部 慎一朗

Lucene や Solr では、ユーザのクエリに対してインデックス内のドキュメントを検索し、検索語にヒットしたドキュメントをユーザに提示する。その際、「ランキング」と呼ばれるドキュメントの順位付けを行い適切に並び替えを行う。「ランキング」が適切になされることにより、ユーザは探していたドキュメントを素早く探し求めることができる。ランキングはユーザクエリとヒットした各ドキュメントとの類似度（関連度）を数学的に計算して行う。ユーザクエリとヒットしたひとつのドキュメントの類似度の計算を本書では「スコア計算」と呼ぶ。Lucene ではブール代数モデルとベクトル空間モデルをカスタマイズして組み合わせて使っているが、このスコア計算は Similarity クラスというところで行われている。

その計算内容については Similarity クラスの Javadoc で説明されているが、Lucene 2.9 から非常に読みやすく改訂されている：

Similarity クラスの Javadoc
http://lucene.apache.org/java/2_9_4/api/all/org/apache/lucene/search/Similarity.html

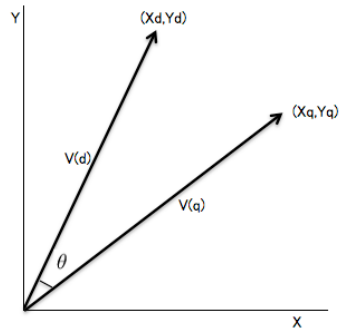
改訂について
https://issues.apache.org/jira/browse/LUCENE-1908

本書では Lucene 流にカスタマイズされたベクトル空間モデルのスコア計算を、Excel を使ってシミュレーションすることを試みる。別紙の Excel をダウンロードして実行しつつ本書をお読みいただきたい。

1. ベクトル空間モデルの基礎

ベクトル空間モデルにおけるスコアは、ユーザクエリとドキュメントの「コサイン類似度」によって求められる。ユーザのクエリに含まれる単語を使い、「タームベクトル V」を形成し、ユーザクエリの V(q)とドキュメントの V(d)のなす角 θ の余弦(cos θ)をスコアとする。

下図のように、2 次元のベクトル（ユーザクエリが 2 単語の場合は 2 次元となる）で説明すれば、ベクトルのなす角の余弦が 1 に近いほど類似しているということになる。



コサイン類似度の公式は次のとおりである。

$$\text{cosine-similarity}(q,d) = \frac{V(q) \cdot V(d)}{|V(q)|\,|V(d)|}$$

ここで V(q)・V(d)は内積、|V(q)||V(d)|はノルムである。この式を表現したのが別紙 Excel の表（1）だ。

このシンプルなモデルには問題がある。次のように

“Apache OR Solr”で検索した場合の、４つのドキュメントのスコア計算をこの式を使ってやってみよう。

クエリ検索語
Apache Solr

No	ドキュメント	V(q)・V(d)	|V(q)||V(d)|	cosine-similarity(q,d)	スコア数値
1	Apache Lucene	1	√2	1/√2	0.707107
2	Apache Solr	2	2	1	1.000000
3	Apache Lucene Solr	2	2	1	1.000000
4	Apache Lucene Solr java	2	2	1	1.000000

すると、ドキュメント２、３、４のスコアが同値になってしまう。ドキュメント全体から考えれば、ドキュメント２のスコアが最も高くなるべきである。この問題は、ドキュメント２、３、４のノルムがみな同じになってしまっていることから発生している。この公式では、ドキュメントの単語数による長さが考慮されていない。

2. 「Lucene 流」スコア計算

～ 概念モデル ～

そこで Lucene では、上記の問題を回避しつつ、スコア計算の精度を高めるために、次のように計算式をカスタマイズしている（スコア概念モデル）。

$$score(q,d) = coord\text{-}factor(q,d) \cdot query\text{-}boost(q) \cdot \frac{V(q) \cdot V(d)}{|V(q)|} \cdot doc\text{-}len\text{-}norm(d) \cdot doc\text{-}boost(d)$$

この式ではドキュメントのノルムを外に出して、次の項目に置き換えていることがわかる。

項目	説明
doc-len-norm(d)	ドキュメントの単語数の長さを考慮する係数 doc-len-norm=1/√単語数
doc-boost(d)	特定ドキュメントに対して恣意的に重み付ける係数

前述と同じ４つのドキュメントに対し、スコア概念モデルの doc-len-norm を加味して計算しているのが別紙 Excel の表（２）である。

クエリ検索語
Apache Solr

No	ドキュメント	V(q)・V(d)	|V(q)|	doc-len-norm	score(q,d)	スコア数値
1	Apache Lucene	1	√2	1/√2	1/√2・1/√2	0.500000
2	Apache Solr	2	√2	1/√2	2/√2・1/√2	1.000000
3	Apache Lucene Solr	2	√2	1/√3	2/√2・1/√3	0.816497
4	Apache Lucene Solr java	2	√2	1/√4	2/√2・1/√4	0.707107

その結果、ドキュメント２のスコアが最も高くなるように改善されたことがわかる。

なお、Lucene のスコア概念モデルでは次の項目も定義しているが、簡単化のため Excel の表では省略している。

項目	説明
coord-factor (q,d)	ドキュメントの中で一致するクエリタームの数を考慮する係数
query-boost (q)	特定クエリタームに対して恣意的に重み付ける係数

3. 「Lucene 流」スコア計算

～ 実践モデル ～

さて、Lucene の実際のスコア計算では、前述のスコア概念モデルに対し、tf-idf の重み付けを考慮して再定義している。tf-idf とは、ドキュメントやクエリ内で単語が出現する頻度（tf）と、インデックス内における単語の稀少価値（idf）によって単語を重み付けるアルゴリズムである。この結果、スコア精度をより高めた Lucene のスコア実践モデルが提供される。次のモデル公式が最終的なスコア計算式となる。

$$score(q,d) = coord(q,d) \cdot queryNorm(q) \cdot \sum_{t \text{ in } q} \left(tf(t \text{ in } d) \cdot idf(t)^2 \cdot t.getBoost() \cdot norm(t,d) \right)$$

上記公式では、次の項目を積上げたものからスコアが計算できると定義されている。

項目	説明
coord(q,d)	ドキュメントの中で一致するクエリタームの数を考慮する係数
queryNorm(q)	異なるクエリ間でスコアを比較できるようにするための正規化係数
tf(t in d)	ドキュメント(d)の中での単語(t)の頻度
idf(t)	インデックス内の単語(t)の希少価値
t.getBoost()	検索クエリの中で特定クエリタームを重み付ける係数
norm(t,d)	特定ドキュメントを重み付ける係数、特定フィールドを重み付ける係数、ドキュメントの単語数の長さを考慮する係数、をまとめた係数

Lucene のスコア実践モデルでは、Lucene のスコア概念モデルから、ドキュメントのノルムをまとめて「norm(t,d)」に置き換えている。また、V(q)・V(d)の部分は、重み係数を 1 から tf-idf に置き換えている。「Σ tf(t in d)idf(t)^2」の部分は 2 次元の場合、次のように説明できる。

$$
\begin{aligned}
&V(q)\cdot V(d)\\
&=(vq_1, vq_2) (vd_1, vd_2)^T\\
&=vq_1*vd_1+vq_2*vd_2\\
&=tf(q_1)idf(t)*tf(d_1)idf(t)+tf(q_2)idf(t)*tf(d_2)idf(t)\\
&=tf(t \text{ in } q_1)idf(t)*tf(t \text{ in } d_1)idf(t)\\
&\quad +tf(t \text{ in } q_2)idf(t)*tf(t \text{ in } d_2)idf(t)\\
&=idf(t)*tf(t \text{ in } d_1)idf(t) + idf(t)*tf(t \text{ in } d_2)idf(t) \quad ※\\
&=idf(t)^2*tf(t \text{ in } d_1) + idf(t)^2*tf(t \text{ in } d_2)\\
&=\sum_t (tf(t \text{ in } d)*idf(t)^2)
\end{aligned}
$$

※ tf(t in q)は常に 1 なので省略。

tf と idf の具体的な計算方法は抽象クラス Similarity の具象クラスである DefaultSimilarity にて次のとおり実装されている。

項目	説明
tf	1 ドキュメントの中で単語がどれだけ頻繁に出現するかを表す。多く出現するほど高い数値となる。tf = 出現頻度^(1/2)
idf	全ドキュメントの中で単語がどれだけ稀かを表す。稀であるほど高い数値となる。インデックス内のドキュメント総数が N で単語 t を含むドキュメントの数を df とすると、idf(t)=log(N/(df+1))+1 となる。たとえば、すべての文書に対象単語が出現する場合は、限りなく 1 に近くなり、ほとんど出現しない場合は大きな値になる。

前述の 4 つのドキュメントを、スコア実践モデルの tf、idf、norm を加味して計算しているのが別紙 Excel の表（3）である。

クエリ検索語
Apache Solr

		t="Apache"			t="Solr"			
No	ドキュメント	tf(t)	idf(t)^2	norm(t)	tf(t)	idf(t)^2	norm(t)	スコア数値
1	Apache Lucene	1^(1/2)	1+LOG(4/4+1)^2	1/√2				0.576696
2	Apache Solr	1^(1/2)	1+LOG(4/4+1)^2	1/√2	1^(1/2)	1+LOG(4/3+1)^2	1/√2	1.283803
3	Apache Lucene Solr	1^(1/2)	1+LOG(4/4+1)^2	1/√3	1^(1/2)	1+LOG(4/3+1)^2	1/√3	1.048221
4	Apache Lucene Solr java	1^(1/2)	1+LOG(4/4+1)^2	1/√4	1^(1/2)	1+LOG(4/3+1)^2	1/√4	0.907786

今回の簡単なサンプルでは、ドキュメント数とドキュメント内の単語数も少ないため、tf-idf の重み付けの価値がはっきりわからないが、実際の場面では威力を発揮するはずである。

4. おわりに

本書では、全文検索のスコア計算におけるベクトル空間モデルの基礎を最初に説明し、Lucene 流にいかにカスタマイズしているかを解説した。別紙 Excel ファイルとともに本書をお読みいただき、Lucene や Solr のスコア計算について理解を深めていただければ幸いである。

（株）ロンウイットについて
ロンウイットはオープンソースの全文検索エンジン Lucene /Solr を企業システムに導入する支援サービス事業を展開している。
お問い合わせ先
〒100-0005 東京都千代田区丸の内 1-8-3
丸の内 トラストタワー本館 20 階
電話：03-5288-5927　　FAX：03-5288-5928
メール：sales@rondhuit.com
ホームページ：http://www.rondhuit.com/