Substring Search

Contents of this Page

Substring Algorithm Implementation Details

Search based on a \"*some sub-string here*\". Functions much like the Java String.indexOf method. This requires two indexed fields to manage this without significant overhead. One field is the tokenized property value which causes no extra indexing, the other is reversed which requires an extra indexed field.

When multiple terms are being searched on, the first term is a spanWildcardQuery on the reverse property with a trailing wildcard. The middle property values are searched for as property values. The last term is a spanWildcardQuery on the propertyValue with a tailing wildcard.

Algorithm:

The Substring search has the following characteristics:

- This search is case in-sensitive.
- It searches on the property value and literal property value.
- The literal property part of the query is boosted by 50. This gives a literal match priority.
- Performs a wildcardQuery
- Lowercase and special characters removed during query parser parse.
- Parsing is done with the following analyzers:
  - propertyValue - Uses our custom standard analyzer that has no stop words.
  - literal_propertyValue - Uses our custom literal analyzer. This literal analyzer uses Lucene's WhitespaceTokenizer with Lucene's LowerCaseFilter.

Example of use:

The following examples are based on the Automobiles coding scheme.

Example 1:

Search string: graph

Lucene query: +propertyValue:*graph* literal_propertyValue:graph^50.0

Result: 1 result

- entity code: NoRelationsConcept
- entity description: A concept for testing **Graph** Building on Concepts with no relations

Example 2:
Search string: graph building on

Lucene query: +spanNear([mask(spanWildcardQuery(reverse_propertyValue:hparg*)) as propertyValue, mask(propertyValue:building) as propertyValue, mask(spanWildcardQuery(propertyValue:on*)) as propertyValue], 0, true) ((+literal_propertyValue:graph +literal_propertyValue:building +literal_propertyValue:on)^50.0)

Result: 1 result

- entity code: NoRelationsConcept
- entity description: A concept for testing **Graph** Building on Concepts with no relations

Example 3:
Search string: ncept for testing graph

Lucene query: +spanNear([mask(spanWildcardQuery(reverse_propertyValue:tpecn*)) as propertyValue, mask(propertyValue:for) as propertyValue, mask(propertyValue:testing) as propertyValue, mask(spanWildcardQuery(propertyValue:graph*)) as propertyValue], 0, true) ((+literal_propertyValue:ncept +literal_propertyValue:for +literal_propertyValue:testing +literal_propertyValue:graph)^50.0)

Result: 1 result

- entity code: NoRelationsConcept
- entity description: A concept for testing **Graph** Building on Concepts with no relations

Associated JUnits:

Junit tests can be found here: https://github.com/lexevs/lexevs/blob/master/lbTest/src/test/java/org/LexGrid/LexBIG/Impl/function/query/lucene/searchAlgorithms/TestSubString.java