Contains Search

Contents of this Page

Contains Algorithm Implementation Details

Equivalent to ' term* ' - in other words - a trailing wildcard on a term (but no leading wild card) and the term can appear at any position.

Algorithm:

The contains search has the following characteristics:

- This search is case in-sensitive.
- It only 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.
- A trailing wild card is added to all tokens in the search text.
- 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: automob

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

Result: 1 result

- entity code: A0001
- entity description: Automobile

Example 2:
Search string: General Motors

Lucene query: (+propertyValue:general* +propertyValue:motors*) ((+literal_propertyValue:general +literal_propertyValue:motors)^50.0)

Result: 1 result

- entity code: GM
- entity description: General Motors

Associated JUnits:

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