Starts With Search

Contents of this Page

Starts With 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 Starts With search has the following characteristics:

- This search is case in-sensitive.
- It searches on the untokenizedLCPropertyValue and the property value.
- The literal property part of the query is boosted by 50. This gives a literal match priority.
- A trailing wild card is on the term (but no leading wild card) and the term can appear at any position.
- Lowercase and special characters removed during query parser parse.
- Parsing is done with the following analyzers:
  - untokenizedLCPropertyValue - Analyzers are not applied to property value. However, the expression is lower cased (this is an explicit step done outside of Lucene by LexEVS code).
  - 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: +untokenizedLCPropertyValue:automob* literal_propertyValue:automob^50.0

Result: 1 result

- entity code: A0001
- entity description: Automobile

Example 2:

Search string: Car (with special) charaters!

Lucene query: +untokenizedLCPropertyValue:car (with special) charaters!* ((+literal_propertyValue:car +literal_propertyValue:(with +literal_propertyValue: special) +literal_propertyValue:charaters!)^50.0)

Result: 1 result

- entity code: C0001
- entity description: Car

Associated JUnits:

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