Leading and Trailing Wild Card Search

Contents of this Page

Leading and Trailing Wild Card Implementation Details

Equivalent to '*term*' This should be a very poor performing search and is not recommended especially when entering a phrase.

Algorithm:

The Leading and Trailing Wild Card search has the following characteristics:

- This search is case in-sensitive.
- It only searches on the property value and literal property value.
- A leading and trailing wild card is added to each token in the search text.
- The literal property part (without the wild cards) of the query is boosted by 50. This gives a literal match priority.
- 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: hevy

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

Result: 1 result

- entity code: Chevy
- entity description: Chevrolet

Example 2:
Search string: hev

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

Result: 1 result

- entity code: Chevy
- entity description: Chevrolet

Associated JUnits:

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