Spelling Error Tolerant Substring Search

Contents of this Page

Spelling Error Tolerant Substring Implementation Details

Adds Spelling-error tolerance to 'subString' search. This makes use of the double metaphone indexed value as well as literal property values.

Algorithm:

The Spelling Error Tolerant Substring search has the following characteristics:

- This search is case in-sensitive.
- It searches on the double metaphone property value and literal property value.
- The literal property part (without the wild cards) of the query is boosted by .5. This gives a literal match priority.
- Parsing is done with the following analyzers:
  - dm_propertyValue - Uses our custom double metaphone analyzer. This has the following filters:
    - LowerCaseFilter - for setting to lowercase
    - StopFilter - to remove stop words (the, a, etc.) from the search
    - DoubleMetaphoneFilter - for testing double metaphone sounds
  - 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: car

Lucene query: dm_propertyValue:"KR" literal_propertyValue:"car"^0.5

Result: 2 results

- Result 1
  - entity code: C0001
  - entity description: Car
- Result 2
  - entity code: C0002
  - entity description: Kar

Example 2:
Search string: General Motors

Lucene query: dm_propertyValue:"JNRL KNRL MTRS" literal_propertyValue:"general motors"^0.5

Result: 1 result

- entity code: GM
- entity description: General Motors

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/TestSpellingErrorTolerantSubString.java