Minimum Snippet

Project Number 1

Generated by Doxygen 1.8.9.1

Sat Aug 29 2015 20:13:26

Contents

1 Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

2 Class Documentation

2.1 MinimumSnippet Class Reference

Public Member Functions

- MinimumSnippet (Iterable< String > document, List< String > terms)
- boolean foundAllTerms ()
- int getStartingPos ()
- int getEndingPos ()
- int getLength ()
- int getPos (int index)

2.1.1 Detailed Description

When you do a web search, the results page shows you a `snippet` for each result, showing you search terms in context. For purposes of this project, a snippet is a subsequence of a document that contains all the search terms.

For this project, you will write code that, given a document (a sequence of words) and set of search terms, finds the minimal length subsequence in the document that contains all of the search terms.

If there are multiple subsequences that have the same minimal length, you may return any one of them.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 MinimumSnippet (Iterable< String > *document,* List< String > *terms*)

Compute minimum snippet.

Given a document (represented as an List<String>), and a set of terms (represented as a List<String>), find the shortest subsequence of the document that contains all of the terms.

This constructor should find the minimum snippet, and store information about the snippet in fields so that the methods can be called to query information about the snippet. All significant computation should be done during construction.

@param document The Document is an Iterable<String>. Do not change the document. It is an Iterable, rather than a List, to allow for implementations where we scan very large documents that are not read entirely into memory. If you have problems figuring out how to solve it with the document represented as an Iterable, you may cast it to a List<String>; in all but a very small number of test cases, in will in fact be a List<String>.

@param terms The terms you need to look for. The terms will be unique (e.g., no term will be repeated), although you do not need to check for that. There should always be at least one term and your code should throw an `IllegalArgumentException` if "terms" is empty.

2.1.3 Member Function Documentation

2.1.3.1 boolean foundAllTerms ( )

Returns whether or not all terms were found in the document. If all terms were not found, then none of the other methods should be called.

@return whether all terms were found in the document.

2.1.3.2 int getEndingPos ( )

Return the ending position of the snippet

@return the index in the document of the last element of the snippet

2.1.3.3 int getLength ( )

Return total number of elements contained in the snippet.

@return an integer greater than or equal to 1

2.1.3.4 int getPos (int *index*)

Returns the position of one of the search terms as it appears in the original document

@param index index of the term in the original list of terms. For example, if index is 0 then the method will return the position (in the document) of the first search term. If the index is 1, then the method will return the position (in the document) of the second search term. Etc.

@return position of the term in the document

2.1.3.5 int getStartingPos ( )

Return the starting position of the snippet

@return the index in the document of the first element of the snippet

The documentation for this class was generated from the following file:

- src/student_classes/MinimumSnippet.java

Index