7/1/2020

Sourabh Roy

**INFORMATION RETRIEVAL API BASED ON NLP TECHNIQUES**

**API Guidelines**

This is a Information Retrieval Library where you can use built in data types and functions that can be used to tokenize documents, find Jaccard similarity, Cosine similarity and transform using TF-IDF.

Library is under **namespace aether**

**Example**

#include <nlpirobj.h>

int main( )

{

aether :: word\_to\_vec obj;

obj = “Hello world”;

std :: string b = " hello";

aether :: word\_to\_vec cdf = b;

std :: cout << cdf <<’\n’;

aether :: wordbag mat = aether :: generate\_wordbag( obj , cdf );

std :: cout << mat;

std :: cout << “\nJaccard similarity : ” << mat.jaccard();

return 0;

}

**Class Word\_to\_vec** contains

This class accepts a string and uses tokenizing function. It also transform multiple documents to form a bag of word matrix.

**MEMBER VARIABLE**

|  |  |  |
| --- | --- | --- |
| std::string text | Contains built in C++ string type | ( C++11 ) |

This variable “text” holds a document in private scope and can be used or modified using proper function only. Not for direct use.

**MODIFYING OPERATION**

|  |  |  |
| --- | --- | --- |
| word\_to\_vec () | Class constructor | ( C++11 ) |
| word\_to\_vec ( const std::string& ) | Class construction with string reference | ( C++11 ) |
| void operator = ( const std::string& ) | Operator overloading for string assignment | ( C++11 ) |
| ~word\_to\_vec () | Object destructor | ( C++11 ) |

**Constructor word\_to\_vec**

Initializes word\_to\_vec object with a string assigned by reference.

**Example**

word\_to\_vec obj ( “Harappa civilisation is a epitome of our ancient culture.” );

**Function operator =**

Overloaded function = can be used to assign values to initialize objects of type word\_to\_vec.

Return type -> void

Time complexity -> O( n ) where n is the length of the document.

Space complexity -> O( n ) where n is the size of the document.

**Example**

word\_to\_vec obj , obj2;

string file1 = “Harappa civilisation is a epitome of our ancient culture.”;

obj = “Harappa civilisation is a epitome of our ancient culture.”; // overloaded = function

obj2 = file1; // overloaded = function

**NON MODIFYING OPERATION**

|  |  |  |
| --- | --- | --- |
| friend std::ostream& operator << ( std::ostream& i, const word\_to\_vec& j ) | Outstream operator | ( C++11 ) |
| friend std::unordered\_map < std::string , std::pair < long double, long double > > generate\_wordbag ( const word\_to\_vec& , const word\_to\_vec& ) | Friend word bag generating operation | ( C++11 ) |

**Function operator <<**

It can be used to output abstract data type to print contents of word\_to\_vec object contents. It is used to show tokenized vector representation of document. A way to see contents of private data member without calling directly outside the scope but with using this function we can print the contents.

Return type -> Referenced ostream object

Time complexity -> O( n ) where n is the number of rows in word bag matrix.

Space complexity -> O( 1 ) as function parameter is referenced.

**Example**

word\_to\_vec obj; // Object declaration of type word\_to\_vec

obj = ”Hello world”; // Object initialisation

cout << obj; // Prints contents of object obj.

**Function generate\_wordbag**

Calculates word bag matrix representation of two document vector already tokenized.

Return type -> unordered\_map < std::string , std::pair < long double, long double > >

Time complexity -> O( n ) where n is the max length among two documents .

Space complexity -> O( 1 ) as function parameter is referenced.

**Example**

auto bag\_of\_words = generate\_wordbag ( file , file1 ); // returns a single matrix word bag of two

// documents

**Class Wordbag** contains

This class contains information retrieval techniques and transformation. Object of this class can hold a bag of words matrix on which several operation can be used.

**MEMBER VARIABLE**

|  |  |  |
| --- | --- | --- |
| std::unordered\_map < std::string , std::pair < long double , long double > > \_a | Contains hashmap matrix type | ( C++11 ) |

It holds a word bag matrix representation of two documents in private scope. It may be used to find similarity between two documents using techniques provided.

**MODIFYING OPERATION**

|  |  |  |
| --- | --- | --- |
| wordbag (const std::unordered\_map < std::string , std::pair < long double, long double > >& ) const | Class constructor | ( C++11 ) |
| ~wordbag () | Object destructor | ( C++11 ) |
| void operator = ( const std::unordered\_map < std::string , std::pair < long double, long double > >& ) | Operator overloading for hashmap of matrix assignment | ( C++11 ) |

**Constructor wordbag**

Helps to initialize wordbag objects by assigning a word bag matrix.

**Example**

Auto mat = vec.generate\_wordbag( file1 , file2 );

Wordbag obj ( mat );

**Function operator =**

Overloaded = function to initialize wordbag object with word bag matrix.

Return type -> unordered\_map < std::string , std::pair < long double, long double > >

Time complexity -> O( n ) where n is the number of rows of word bag matrix.

Space complexity -> O( n ) where n is the size of the matrix.

**Example**

wordbag obj;

obj = vec.generate\_wordbag( file1 , file2 ); // overloaded = function

**NON MODIFYING OPERATION**

|  |  |  |
| --- | --- | --- |
| double cosine () const | Cosine similarity operation | ( C++11 ) |
| double jaccard () const | Jaccard similarity operation | ( C++11 ) |
| std::unordered\_map < std::string , std::pair < long double, long double > > transform\_tfidf () const | TFIDF transformation operation | ( C++11 ) |
| friend std::ostream& operator << ( std::ostream& , const wordbag& ) | Outstream operator | ( C++11 ) |

**Function cosine**

Finds plain vector cosine similarity between two documents.

Return type -> int

Time complexity -> O( n ) where n is the number of rows of word bag matrix.

Space complexity -> O( 1 )

**Example**

wordbag obj = vec.generate\_wordbag( file1 , file2 );

int result = obj.cosine();

**Function jaccard**

Finds plain vector jaccard similarity between two documents.

Return type -> int

Time complexity -> O( n ) where n is the number of rows of word bag matrix.

Space complexity -> O( 1 )

**Example**

wordbag obj = vec.generate\_wordbag( file1 , file2 );

int result = obj.jaccard( );

**Function operator <<**

Overloaded << function to print contents of wordbag object using ostream object.

Return type -> unordered\_map < std::string , std::pair < long double, long double > >

Time complexity -> O( n ) where n is the number of rows of word bag matrix.

Space complexity -> O( n ) where n is the size of the matrix.

**Example**

wordbag obj = vec.generate\_wordbag( file1 , file2 );

cout << obj; //overloaded << operator

**Function transform\_tfidf**

Computes term frequency and inverse term frequency of a given word bag matrix and returns the transformed matrix.

Return type -> unordered\_map < std::string , std::pair < long double, long double > >

Time complexity -> O( n ) where n is the number of rows of word bag matrix.

Space complexity -> O( n ) where n is the size of the matrix.

**Example**

wordbag obj = vec.generate\_wordbag( file1 , file2 );

wordbag obj\_TFIDF = obj.transform\_tfidf( ); // transforms plain word count matrix