LANGUAGE4

0

Generated by Doxygen 1.9.1

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 Bigram Class Reference

It represents a pair of characters. It is used to store pairs of consecutive characters from a text. It uses a string to store the pair of characters.

`#include <Bigram.h>`

Public Member Functions

- Bigram (const std::string &text="__")

  It builds a Bigram object with `text` as the text of the bigram. If the string `text` contains a number of characters other than two, then the text of the bigram will be initialized with "__".
- Bigram (char first, char second)

  It builds a Bigram object using the two characters passed as parameters of this constructor as the text of the bigram.
- Bigram (const char text[ ])

  It builds a Bigram object with `text` (a c-string) as the text of the bigram. If the c-string `text` contains a number of character other than two, then the text of the bigram will be initialized with "__".
- std::string getText () const

  Obtains a copy of the text of this bigram as a string object.
- std::string toString () const

  Obtains a copy of the text of this bigram as a string object.
- const char & at (int index) const

  Gets a const reference to the character at the given position.
- char & at (int index)

  Gets a reference to the character at the given position.
- void toUpper ()

3.1.1 Detailed Description

It represents a pair of characters. It is used to store pairs of consecutive characters from a text. It uses a string to store the pair of characters.

Definition at line 27 of file Bigram.h.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Bigram() [1/3]

```
Bigram::Bigram (
            const std::string & text = "__" )
```

It builds a Bigram object with `text` as the text of the bigram. If the string `text` contains a number of characters other than two, then the text of the bigram will be initialized with "__".

Parameters

text	the text for the bigram. It should be a string with just two characters.

Definition at line 22 of file Bigram.cpp.

```
22                                          {
23     if (text.size() == 2) {
24         strcpy(_text,text.c_str());
25     }
26     else{
27         strcpy(_text, "__"); // ¿Lanzar excepción?
28     }
29 }
```

3.1.2.2 Bigram() [2/3]

```
Bigram::Bigram (
            char first,
            char second )
```

It builds a Bigram object using the two characters passed as parameters of this constructor as the text of the bigram.

Parameters

first	the first character for the bigram
second	the second character for the bigram

Definition at line 31 of file Bigram.cpp.

```
31                                          {
32     _text[0] = first;
33     _text[1] = second;
34     _text[2] = '\0';
35 }
```

3.1.2.3 Bigram() [3/3]

```
Bigram::Bigram (
            const char text[] )
```

It builds a Bigram object with `text` (a c-string) as the text of the bigram. If the c-string `text` contains a number of character other than two, then the text of the bigram will be initialized with "__".

Parameters

text	the text for the bigram

3.1.3 Member Function Documentation

3.1.3.1 at() [1/2]

```
char & Bigram::at (
            int index )
```

Gets a reference to the character at the given position.

Parameters

index	the position to consider

Exceptions

std::out_of_range	Throws a std::out_of_range exception if the index is not equals to 0 or 1

Returns

A reference to the character at the given position

Definition at line 56 of file Bigram.cpp.

```
56                          {
57     if(index<0 || index>1){
58         throw std::out_of_range(string("char& Bigram::at(int index): ") +
59                 "invalid position " + to_string(index));
60     }
61     else{
62         return _text[index];
63     }
64 }
```

3.1.3.2 at() [2/2]

```
const char & Bigram::at (
            int index ) const
```

Gets a const reference to the character at the given position.

Parameters

index	the position to consider

Exceptions

std::out_of_range	Throws a std::out_of_range exception if the index is not equals to 0 or 1

Returns

A const reference to the character at the given position

Definition at line 46 of file Bigram.cpp.

```
46                                          {
47     if(index<0 || index>1){
48         throw std::out_of_range(string("const char& Bigram::at(int index) const: ") +
49                 "invalid position " + to_string(index));
50     }
51     else{
52         return _text[index];
53     }
54 }
```

3.1.3.3 getText()

```
std::string Bigram::getText ( ) const
```

Obtains a copy of the text of this bigram as a string object.

Returns

The text of this bigram as a string object

Definition at line 66 of file Bigram.cpp.

```
66                                  {
67     return string(_text);
68 }
```

3.1.3.4 toString()

```
std::string Bigram::toString ( ) const
```

Obtains a copy of the text of this bigram as a string object.

Returns

The text of this bigram as a string object

Definition at line 70 of file Bigram.cpp.

```
70                                    {
71     return string(_text);
72 }
```

3.1.3.5 toUpper()

```
void Bigram::toUpper ( )
```

Converts lowercase letters in this bigram to uppercase

Definition at line 74 of file Bigram.cpp.

```
74                          {
75     at(0) = toupper(at(0));
76     at(1) = toupper(at(1));
77 }
```

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

- include/Bigram.h
- src/Bigram.cpp

3.2 BigramFreq Class Reference

A pair of a Bigram object and a frequency (an int), that gives the frequency of a Bigram (times it appears) in a text.

```
#include <BigramFreq.h>
```

Public Member Functions

- BigramFreq ()

  Base constructor. It builds a BigramFreq object with "__" as the text of the bigram and 0 as the frequency.
- const Bigram & getBigram () const

  Gets a const reference to the Bigram of this BigramFreq object.
- int getFrequency () const

  Gets the frequency of this BigramFreq object.
- void setBigram (const Bigram &bigram)

  Sets the Bigram of this BigramFreq object.
- void setFrequency (int frequency)

  Sets the frequency of this BigramFreq object.
- std::string toString () const

  Obtains a string with the string and frecuency of the bigram in this object (separated by a whitespace).

3.2.1 Detailed Description

A pair of a Bigram object and a frequency (an int), that gives the frequency of a Bigram (times it appears) in a text.

Definition at line 27 of file BigramFreq.h.

3.2.2 Member Function Documentation

3.2.2.1 getBigram()

```
const Bigram & BigramFreq::getBigram ( ) const
```

Gets a const reference to the Bigram of this BigramFreq object.

Returns

A const reference to the Bigram of this BigramFreq object

Definition at line 24 of file BigramFreq.cpp.

```
24                                          {
25     return _bigram;
26 }
```

3.2.2.2 getFrequency()

```
int BigramFreq::getFrequency ( ) const
```

Gets the frequency of this BigramFreq object.

Returns

The frequency of this BigramFreq object

Definition at line 27 of file BigramFreq.cpp.

```
27                                      {
28     return _frequency;
29 }
```

3.2.2.3 setBigram()

```
void BigramFreq::setBigram (
            const Bigram & bigram )
```

Sets the Bigram of this BigramFreq object.

Parameters

bigram	The new Bigram value for this object

Definition at line 31 of file BigramFreq.cpp.

```
31                                              {
32     this->_bigram = bigram;
33 }
```

3.2.2.4 setFrequency()

```
void BigramFreq::setFrequency (
            int frequency )
```

Sets the frequency of this BigramFreq object.

Exceptions

std::out_of_range	if `frequency` is negative

Parameters

frequency	the new frequency value for this BigramFreq object

Definition at line 35 of file BigramFreq.cpp.

```
35                                          {
36     if(frequency<0){
37         throw  std::out_of_range(string("void BigramFreq::setFrequency(int frequency): ") +
38                 "invalid frecuency " + to_string(frequency));
39     }
40     this->_frequency = frequency;
41 }
```

3.2.2.5 toString()

```
string BigramFreq::toString ( ) const
```

Obtains a string with the string and frecuency of the bigram in this object (separated by a whitespace).

Returns

A string with the string and frecuency of the bigram in this object

Definition at line 43 of file BigramFreq.cpp.

```
43                                      {
44     return _bigram.toString() + " " + to_string(_frequency);
45 }
```

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

- include/BigramFreq.h
- src/BigramFreq.cpp

3.3 Language Class Reference

It defines a model for a given language. It contains a vector of pairs Bigram-frequency (objects of the class BigramFreq) and an identifier (string) of the language.

```
#include <Language.h>
```

Public Member Functions

- Language ()

  Base constructor. It builds a Language object with "unknown" as identifier, and an empty vector of pairs Bigram-frequency.
- Language (int numberBigrams)

  It builds a Language object with "unknow" as identifier, and a vector of `numberBigrams` pairs Bigram-frequency. Each pair will be initialized as "__" for the Bigram and 0 for the frequency.
- Language (const Language &orig)

  Copy constructor.
- ~Language ()

  Destructor of class Language.
- Language & operator= (const Language &orig)

  Overloading of the assignment operator for Language class.
- const std::string & getLanguageId () const

  Returns the identifier of this language object.
- void setLanguageId (const std::string &id)

  Sets a new identifier for this language object.
- const BigramFreq & at (int index) const

  Gets a const reference to the BigramFreq at the given position of the vector in this object.
- BigramFreq & at (int index)

  Gets a reference to the BigramFreq at the given position of the vector in this object.
- int getSize () const

  Gets the number of BigramFreq objects.
- double getDistance (const Language &otherLanguage) const

  Gets the distance between this Language object (L_1) and the given one `otherLanguage` (L_2). The distance between two Languages L_1 and L_2 is calculated in the following way: $d = \sum_{bigram_i(L_1)} \frac{|rank^{L_1}_{bigram_i(L_1)} - rank^{L_2}_{bigram_i(L_1)}|}{size(L_1) * size(L_1)}$, where $bigram_i(L_j)$ is the bigram i of the Language $L_j, j \in \{1,2\}$ and $rank^{L_k}_{bigram_i(L_j)}$ is the ranking of the bigram i of the Language $L_j, j \in \{1,2\}$ in the Language L_k. The rank of a bigram is the position in which it appears in the list of BigramFreq. We consider 0 as the first position (rank equals to 0). When calculating $rank^{L_2}_{bigram_i(L_1)}$, if the bigram $bigram_i(L_1)$ does not appears in the Language L_2 we consider that the rank is equals to the size of Language L_1.
- int findBigram (const Bigram &bigram) const

  Searchs the given bigram in the list of bigrams in this Language. If found, it returns the position where it was found. If not, it returns -1. We consider that position 0 is the one of the first bigram in the list of bigrams and this->getSize()-1 the one of the last bigram.
- std::string toString () const

  Obtains a string with the following content:
- void sort ()

  Sort the vector of BigramFreq in decreasing order of frequency. If two BigramFreq objects have the same frequency, then the alphabetical order of the bigrams of those objects will be considered (the object with a bigram that comes first alphabetically will appear first) Modifier method.
- void save (const char fileName[ ]) const

  Saves this Language object in the given file.
- void load (const char fileName[ ])

  Loads into this object the Language object stored in the given file.
- void append (const BigramFreq &bigramFreq)

  Appends a copy of the given BigramFreq to this Language object. If the bigram is found in this object, then its frequency is increased with the one of the given BigramFreq object. If not, a copy of the given BigramFreq object is appended to the end of the list of BigramFreq objects in this Language.
- void join (const Language &language)

  Appends to this Language object, the list of pairs bigram-frequency contained in `language`.

3.3.1 Detailed Description

It defines a model for a given language. It contains a vector of pairs Bigram-frequency (objects of the class BigramFreq) and an identifier (string) of the language.

Definition at line 28 of file Language.h.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Language() [1/2]

```
Language::Language (
            int numberBigrams )
```

It builds a Language object with "unknow" as identifier, and a vector of `numberBigrams` pairs Bigram-frequency. Each pair will be initialized as "__" for the Bigram and 0 for the frequency.

Exceptions

std::out_of_range	Throws a std::out_of_range exception if `numberBigrams` <0

Parameters

numberBigrams	The number of bigrams to use in this Language

Definition at line 32 of file Language.cpp.

```
32                                      {
33     if(numberBigrams < 0){
34         throw std::out_of_range(string("Language::Language(int numberBigrams): ") +
35                 "invalid numberBigrams=" + to_string(numberBigrams));
36     }
37     _languageId="unkown";
38     allocate(numberBigrams);
39 }
```

3.3.2.2 Language() [2/2]

```
Language::Language (
            const Language & orig )
```

Copy constructor.

Parameters

orig	the Language object used as source for the copy

Definition at line 41 of file Language.cpp.

```
41                                      {
42     allocate(orig._size);
43     copy(orig);
44 }
```

3.3.3 Member Function Documentation

3.3.3.1 append()

```
void Language::append (
            const BigramFreq & bigramFreq )
```

Appends a copy of the given BigramFreq to this Language object. If the bigram is found in this object, then its frequency is increased with the one of the given BigramFreq object. If not, a copy of the given BigramFreq object is appended to the end of the list of BigramFreq objects in this Language.

Parameters

bigramFreq	The BigramFreq to append to this object

Definition at line 231 of file Language.cpp.

```
231                                                 {
232     this->reallocate(this->_size + 1);
233 //    this->_vectorBigramFreq[this->size-1] = bigramFreq;
234     this->at(this->_size-1) = bigramFreq;
235 }
```

3.3.3.2 at() [1/2]

```
BigramFreq & Language::at (
            int index )
```

Gets a reference to the BigramFreq at the given position of the vector in this object.

Parameters

index	the position to consider

Exceptions

std::out_of_range	Throws an std::out_of_range exception if the given index is not valid

Returns

A reference to the BigramFreq at the given position

Acceso seguro

Definition at line 76 of file Language.cpp.

```
76                                  {
77     if (0 <= index && index < getSize())
78         return _vectorBigramFreq[index];
79     else
80         throw std::out_of_range(string("BigramFreq& Language::at(int index): ") +
81                 "invalid position " + to_string(index));
82 }
```

3.3.3.3 at() [2/2]

```
const BigramFreq & Language::at (
            int index ) const
```

Gets a const reference to the BigramFreq at the given position of the vector in this object.

Parameters

index	the position to consider

Exceptions

std::out_of_range	Throws an std::out_of_range exception if the given index is not valid

Returns

A const reference to the BigramFreq at the given position

Definition at line 67 of file Language.cpp.

```
67                                            {
68     if (0 <= index && index < getSize())
69         return _vectorBigramFreq[index];
70     else
71         throw std::out_of_range(
72                 string("const BigramFreq& Language::at(int index) const: ") +
73                 "invalid position " + to_string(index));
74 }
```

3.3.3.4 findBigram()

```
int Language::findBigram (
            const Bigram & bigram ) const
```

Searchs the given bigram in the list of bigrams in this Language. If found, it returns the position where it was found. If not, it returns -1. We consider that position 0 is the one of the first bigram in the list of bigrams and this->getSize()-1 the one of the last bigram.

Parameters

bigram	A bigram

Precondition

The list of bigrams should be ordered in decreasing order of frequency. This is not checked in this method.

Returns

If found, it returns the position where the bigram was found. If not, it returns -1

Definition at line 105 of file Language.cpp.

```
105                                                 {
106     for (int i = 0; i < getSize(); ++i) {
107         if (bigram.getText() == this->at(i).getBigram().getText()) { // o bien if (bigrama ==
        (*this)[i].getBigram()) {
108             return i;
109         }
110     }
111     return -1;
112 }
```

3.3.3.5 getLanguageId()

```
const string & Language::getLanguageId ( ) const
```

Returns the identifier of this language object.

Returns

A const reference to the identifier of this language object

Definition at line 59 of file Language.cpp.

```
59                                          {
60     return _languageId;
61 }
```

3.3.3.6 getSize()

```
int Language::getSize ( ) const
```

Gets the number of BigramFreq objects.

Returns

The number of BigramFreq objects

Definition at line 84 of file Language.cpp.

```
84                          {
85     return _size;
86 };
```

3.3.3.7 join()

```
void Language::join (
            const Language & language )
```

Appends to this Language object, the list of pairs bigram-frequency contained in `language`.

Parameters

language	A Language object

Definition at line 237 of file Language.cpp.

```
237                                         {
238     int pos;
239     for(int i=0; i<language.getSize(); i++){
240         pos = this->findBigram(language.at(i).getBigram().getText());
241         if(pos>=0){ // If found
242             this->at(pos).setFrequency(this->at(pos).getFrequency() +
243                     language.at(i).getFrequency());
244         }
245         else{ // If not found
246             this->append(language.at(i));
247         }
248     }
249 }
```

3.3.3.8 load()

```
void Language::load (
              const char fileName[] )
```

Loads into this object the Language object stored in the given file.

Parameters

fileName	A c-string with the name of the file where the Language will be stored.

Exceptions

std::out_of_range	Throws a std::out_of_range exception if the number of bigrams read from the given file is negative.
std::ios_base::failure	Throws a std::ios_base::failure exception if the given file cannot be opened or if an error occurs while reading from the file
throw	std::invalid_argument Throws a std::invalid_argument exception if an invalid magic string is found in the given file

Definition at line 176 of file Language.cpp.

```
176                                             {
177     string bigramString;
178     int frequency;
179     ifstream inputStream;
180     inputStream.open(fileName, ifstream::in ); // antes ifstream flujo(fichero);
181
182     string magicString;
183     if (inputStream) {
184         inputStream » magicString;
185         if (magicString ==  Language::MAGIC_STRING_T) { //Para ficheros texto
186             int nBigrams;
187             string languageId;
188
189             inputStream » languageId;
190             this->setLanguageId(languageId);
191
192             inputStream » nBigrams;
193             if (nBigrams < 0) {
194                 throw std::out_of_range(
195                         string("void Language::load(const char *fileName): ") +
196                         "invalid number of bigrams=" + to_string(_size));
```

```
197             }
198
199             this->deallocate();
200             this->allocate(nBigrams);
201
202             for (int i = 0; i < this->getSize(); ++i) {
203                 inputStream » bigramString;
204                 this->at(i).setBigram(Bigram(bigramString));
205                 inputStream » frequency;
206                 this->at(i).setFrequency(frequency);
207             }
208         }
209         else{
210             throw std::invalid_argument(
211                     string("void Language::load(const char *fileName): ") +
212                     "the found magic string " + magicString + " in file " +
213                     fileName + " is not valid ");
214         }
215         if (inputStream) {
216             inputStream.close();
217         }
218         else{
219             throw std::ios_base::failure(
220                 string("void Language::load(const char *fileName): ") +
221                         "error reading from file " + fileName);
222         }
223     }
224     else{
225         throw std::ios_base::failure(
226                 string("void Language::load(const char *fileName): ") +
227                         "error opening file " + fileName);
228     }
229 }
```

3.3.3.9 operator=()

```
Language & Language::operator= (
            const Language & orig )
```

Overloading of the assignment operator for Language class.

Parameters

orig	the Language object used as source for the assignment

Returns

A reference to this object

Definition at line 50 of file Language.cpp.

```
50                                                  {
51     if (this != &orig) {
52         deallocate();
53         allocate(orig._size );
54         copy(orig);
55     }
56     return *this;
57 }
```

3.3.3.10 save()

```
void Language::save (
            const char fileName[] ) const
```

Saves this Language object in the given file.

Parameters

fileName	A c-string with the name of the file where this Language object will be saved

Exceptions

std::ios_base::failure	Throws a std::ios_base::failure exception if the given file cannot be opened or if an error occurs while writing to the file

Definition at line 142 of file Language.cpp.

```
142                                                 {
143     ofstream stream(fileName, ios::out);
144
145     if (stream) {
146         stream « Language::MAGIC_STRING_T « endl;
147         stream « this->toString();
148         /*stream « getLanguageId() « endl;
149         stream « getSize() « endl;
150         for (int i = 0; i < getSize(); ++i) {
151             stream «  at(i).toString() « endl;
152         }*/
153
154
155         if (stream)  {
156             stream.close();
157 //            return true;
158         }
159         else{
160             throw std::ios_base::failure(
161                 string("void Language::save(const char *fileName, char mode) const: ") +
162                         "error writing to file " + fileName);
163         }
164 //        cerr « "ERROR guardando datos en el fichero " « fileName « endl;
165 //        return false;
166     }
167     else{
168         throw std::ios_base::failure(
169                 string("void Language::save(const char *fileName, char mode) const: ") +
170                         "error opening file " + fileName);
171     }
172 //    cerr « "ERROR abriendo fichero "« fileName « endl;
173 //    return false;
174 }
```

3.3.3.11 setLanguageId()

```
void Language::setLanguageId (
              const std::string & id )
```

Sets a new identifier for this language object.

Parameters

id	The new identifier

Definition at line 63 of file Language.cpp.

```
63                                                 {
64     _languageId = id;
65 }
```

3.3.3.12 toString()

```
std::string Language::toString ( ) const
```

Obtains a string with the following content:

- In the first line, the number of bigrams in this Language
- In the following lines, each one of the pairs bigram-frequency (separated by a whitespace).

Returns

A string with the number of bigrams and the list of pairs of bigram-frequency in the object

Definition at line 114 of file Language.cpp.

```
114                                     {
115     string outputString = this->getLanguageId() + "\n" +
116             to_string(this->getSize()) + "\n";
117
118     for(int i=0; i<this->getSize(); i++){
119         outputString += this->at(i).toString() + "\n";
120     }
121     return outputString;
122 }
```

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

- include/Language.h
- src/Language.cpp

Chapter 4

File Documentation

4.1 include/Bigram.h File Reference

```
#include <iostream>
#include <string>
```

Classes

- class Bigram

  It represents a pair of characters. It is used to store pairs of consecutive characters from a text. It uses a string to store the pair of characters.

Functions

- bool isValidCharacter (char character, const std::string &validCharacters)

4.1.1 Detailed Description

Author

Silvia Acid Carrillo `acid@decsai.ugr.es`
Andrés Cano Utrera `acu@decsai.ugr.es`
Luis Castillo Vidal `L.Castillo@decsai.ugr.es`

Created on 12 February 2023, 10:40

4.1.2 Function Documentation

4.1.2.1 isValidCharacter()

```
bool isValidCharacter (
            char character,
            const std::string & validCharacters )
```

Checks if the given character is contained in `validCharacters`. That is, if the given character can be consireded as part of a word.

Parameters

character	The character to check
validCharacters	The set of characters that we consider as possible characters in a word. Any other character is considered as a separator.

Returns

true if the given character is contained in `validCharacters`; false otherwise

4.2 include/Language.h File Reference

```
#include <iostream>
#include "BigramFreq.h"
```

Classes

- class Language

  It defines a model for a given language. It contains a vector of pairs Bigram-frequency (objects of the class BigramFreq) and an identifier (string) of the language.

4.2.1 Detailed Description

Author

Silvia Acid Carrillo acid@decsai.ugr.es

Andrés Cano Utrera acu@decsai.ugr.es

Luis Castillo Vidal L.Castillo@decsai.ugr.es

Created on 12 February 2023, 10:40

4.3 src/Bigram.cpp File Reference

```
#include <string>
#include <cstring>
#include "Bigram.h"
```

Functions

- bool **isValidCharacter** (char character, const string &validCharacters)

4.3.1 Detailed Description

Author

Silvia Acid Carrillo acid@decsai.ugr.es
Andrés Cano Utrera acu@decsai.ugr.es
Luis Castillo Vidal L.Castillo@decsai.ugr.es

Created on 12 February 2023, 10:40

4.4 src/BigramFreq.cpp File Reference

```
#include <string>
#include "BigramFreq.h"
```

4.4.1 Detailed Description

Author

Silvia Acid Carrillo acid@decsai.ugr.es
Andrés Cano Utrera acu@decsai.ugr.es
Luis Castillo Vidal L.Castillo@decsai.ugr.es

Created on 12 February 2023, 10:40

4.5 src/Language.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <cstring>
#include "Language.h"
```

4.5.1 Detailed Description

Author

Silvia Acid Carrillo acid@decsai.ugr.es
Andrés Cano Utrera acu@decsai.ugr.es
Luis Castillo Vidal L.Castillo@decsai.ugr.es

Created on 12 February 2023, 10:40