LLaMaPUn C library

Generated by Doxygen 1.8.7

Sun Aug 10 2014 01:20:32

Contents

1 Module Index **1**

1.1 Modules 1

2 Data Structure Index **3**

2.1 Data Structures 3

3 File Index **5**

3.1 File List 5

4 Module Documentation **7**

4.1 DNMlib 7

4.2 The ngram library 8

4.3 Paragraph Discrimination 9

4.4 A Library for Stemming Words 10

4.5 Stopword Library 11

4.6 Unicode Normalizer 12

5 Data Structure Documentation **13**

5.1 dnm_chunk Struct Reference 13

5.2 dnm_iterator Struct Reference 13

5.3 dnm_struct Struct Reference 14

5.4 hash_element_string Struct Reference 14

5.4.1 Detailed Description 14

6 File Documentation **15**

6.1 dnmlib.h File Reference 15

6.1.1 Macro Definition Documentation 16

6.1.1.1 DNM_NORMALIZE_MATH 16

6.1.1.2 DNM_SKIP_CITE 16

6.1.1.3 DNM_SKIP_MATH 16

6.1.2 Enumeration Type Documentation 16

6.1.2.1 dnm_level 16

6.1.3 Function Documentation 16

6.1.3.1 createDNM . . . 16
6.1.3.2 dnmIteratorAddAnnotation . . . 16
6.1.3.3 dnmIteratorHasAnnotation . . . 17
6.1.3.4 dnmIteratorHasAnnotationInherited . . . 17
6.1.3.5 dnmIteratorNext . . . 17
6.1.3.6 dnmIteratorPrevious . . . 17
6.1.3.7 freeDNM . . . 17
6.1.3.8 getDnmChildrenIterator . . . 18
6.1.3.9 getDnmIterator . . . 18
6.1.3.10 getDnmIteratorContent . . . 18
6.2 llamapun_ngrams.h File Reference . . . 18
6.2.1 Function Documentation . . . 18
6.2.1.1 llamapun_get_ngrams . . . 19
6.3 llamapun_para_discr.h File Reference . . . 20
6.3.1 Function Documentation . . . 20
6.3.1.1 llamapun_para_discr_get_bags . . . 20
6.4 stemmer.h File Reference . . . 20
6.4.1 Function Documentation . . . 21
6.4.1.1 close_stemmer . . . 21
6.4.1.2 init_stemmer . . . 21
6.4.1.3 morpha_stem . . . 21
6.5 stopwords.h File Reference . . . 21
6.5.1 Function Documentation . . . 21
6.5.1.1 free_stopwords . . . 21
6.5.1.2 is_stopword . . . 21
6.5.1.3 load_stopwords . . . 21
6.5.1.4 read_stopwords_from_json . . . 22
6.6 unicode_normalizer.h File Reference . . . 22
6.6.1 Function Documentation . . . 22
6.6.1.1 normalize_unicode . . . 22

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

DNMlib 7
The ngram library 8
Paragraph Discrimination 9
A Library for Stemming Words 10
Stopword Library 11
Unicode Normalizer 12

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

dnm_chunk 13
dnm_iterator 13
dnm_struct 14
hash_element_string 14

Chapter 3

File Index

3.1 File List

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

dnmlib.h 15
jsoninclude.h . . . . **??**
llamapun_ngrams.h 18
llamapun_ngrams_old.h . . . . **??**
llamapun_para_discr.h 20
llamapun_senna_pos.h . . . . **??**
llamapun_utils.h . . . . **??**
preprocessor.h . . . . **??**
stemmer.h 20
stopwords.h 21
unicode_normalizer.h 22

Chapter 4

Module Documentation

4.1 DNMlib

Most NLP tools work on plain text. However, the XML structure contains useful information about the structure of a document etc. So one tends to switch back and forth. The purpose of this library is to simplify this switching by providing a DNM (document narrative model) and some tools to iterate over it etc.

4.2 The ngram library

Consists of a single function for finding ngrams so far.

4.3 Paragraph Discrimination

Some tools for experiments concerning paragraph discrimination. Paragraph discrimination refers to the idea that in a mathematical document, paragraphs tend to have certain functions. I.e. they're for example proofs, axioms, definitions, ...

4.4 A Library for Stemming Words

Basically, this library is a small wrapper around the (therefore slightly modified) morpha stemmer from the University of Sussex.

4.5 Stopword Library

A small library which provides a function to check whether a word is a regarded a stopword.

4.6 Unicode Normalizer

provides tools for normalizing unicode to ascii. It uses libiconv.

Chapter 5

Data Structure Documentation

5.1 dnm_chunk Struct Reference

Data Fields

- char ∗ **id**
- xmlNode ∗ **dom_node**
- enum dnm_level **level**
- long **offset_parent**
- long **offset_children_start**
- long **offset_children_end**
- char ∗∗ **annotations**
- size_t **number_of_annotations**
- size_t **annotations_allocated**
- char ∗∗ **inherited_annotations**
- size_t **number_of_inherited_annotations**
- size_t **inherited_annotations_allocated**
- size_t **offset_start**
- size_t **offset_end**

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

- dnmlib.h

5.2 dnm_iterator Struct Reference

Data Fields

- dnmPtr **dnm**
- enum dnm_level **level**
- size_t **pos**
- size_t **start**
- size_t **end**

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

- dnmlib.h

5.3 dnm_struct Struct Reference

Data Fields

- xmlDocPtr **document**
- char ∗ **plaintext**
- struct hash_element_string ∗ **annotation_handle**
- struct dnm_chunk ∗ **para_level**
- struct dnm_chunk ∗ **sent_level**
- struct dnm_chunk ∗ **word_level**
- size_t **size_para_level**
- size_t **size_sent_level**
- size_t **size_word_level**
- size_t **size_plaintext**

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

- dnmlib.h

5.4 hash_element_string Struct Reference

```
#include <dnmlib.h>
```

Data Fields

- char ∗ **string**
- UT_hash_handle **hh**

5.4.1 Detailed Description

string element for uthash

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

- dnmlib.h

Chapter 6

File Documentation

6.1 dnmlib.h File Reference

```
#include <libxml/tree.h>
#include <uthash.h>
#include <string.h>
```

Data Structures

- struct hash_element_string
- struct dnm_struct
- struct dnm_chunk
- struct dnm_iterator

Macros

- #define DNM_NORMALIZE_MATH (1 << 0)
- #define DNM_SKIP_MATH (1 << 1)
- #define DNM_SKIP_CITE (1 << 2)

Typedefs

- typedef struct dnm_struct ∗ **dnmPtr**
- typedef struct dnm_iterator ∗ **dnmIteratorPtr**

Enumerations

- enum dnm_level { **DNM_LEVEL_PARA**, **DNM_LEVEL_SENTENCE**, **DNM_LEVEL_WORD**, **DNM_LEVE↩
L_NONE** }

Functions

- dnmPtr createDNM (xmlDocPtr doc, long parameters)
- void freeDNM (dnmPtr dnm)
- dnmIteratorPtr getDnmIterator (dnmPtr dnm, enum dnm_level level)
- dnmIteratorPtr getDnmChildrenIterator (dnmIteratorPtr it)

- int dnmIteratorNext (dnmIteratorPtr it)
- int dnmIteratorPrevious (dnmIteratorPtr it)
- char $*$ getDnmIteratorContent (dnmIteratorPtr it)
- int dnmIteratorHasAnnotation (dnmIteratorPtr it, const char $*$annotation)
- int dnmIteratorHasAnnotationInherited (dnmIteratorPtr it, const char $*$annotation)
- void dnmIteratorAddAnnotation (dnmIteratorPtr it, const char $*$annotation, int writeIntoDOM, int inheritTo↩ Children)

6.1.1 Macro Definition Documentation

6.1.1.1 #define DNM_NORMALIZE_MATH (1 $<<$ 0)

normalize math tags in document

6.1.1.2 #define DNM_SKIP_CITE (1 $<<$ 2)

skip, i.e. ignore, cite tags in document

6.1.1.3 #define DNM_SKIP_MATH (1 $<<$ 1)

skip, i.e. ignore, math tags in document

6.1.2 Enumeration Type Documentation

6.1.2.1 enum **dnm_level**

the different levels for iterators

6.1.3 Function Documentation

6.1.3.1 **dnmPtr** createDNM (xmlDocPtr *doc,* long *parameters*)

creates a DNM

Example call: createDNM(mydoc, DNM_NORMALIZE_MATH $|$ DNM_SKIP_CITE); Memory has to be freed later using freeDNM

See also

freeDNM

Parameters

doc	a pointer to the DOM
parameters	the parameters

Return values

a	pointer to the new DNM

6.1.3.2 void dnmIteratorAddAnnotation (**dnmIteratorPtr** *it,* const char $*$ *annotation,* int *writeIntoDOM,* int *inheritToChildren*)

adds an annotation to a chunk. Again: A faster way for repeatedly adding one annotation should be implemented.

Parameters

it	An iterator referring to the chunk
annotation	The string representation of the annotation
writeIntoDOM	If non-zero: The annotation is also written into the DOM
inheritToChildren	If non-zero: The annotation is inherited to the child chunks

6.1.3.3 int dnmIteratorHasAnnotation (**dnmIteratorPtr** *it,* const char ∗ *annotation*)

Checks whether a chunk has a certain annotation. Note that a faster way should be implemented, if you want to repeatedly check for one annotation.

Parameters

it	A pointer to an iterator
annotation	The string representation of the annotation

Return values

1	if the chunk has the annotation, 0 otherwise

6.1.3.4 int dnmIteratorHasAnnotationInherited (**dnmIteratorPtr** *it,* const char ∗ *annotation*)

like dnmIteratorHasAnnotation, just for annotations inherited (i.e. annotations from the parent tags)

See also

dnmIteratorHasAnnotation

6.1.3.5 int dnmIteratorNext (**dnmIteratorPtr** *it*)

Make an iterator point to the next chunk

Parameters

it	the iterator that shall be incremented

Return values

0	if the iterator points to the last element already, otherwise 1

6.1.3.6 int dnmIteratorPrevious (**dnmIteratorPtr** *it*)

Make an iterator point to the previous chunk

See also

dnmIteratorNext

6.1.3.7 void freeDNM (**dnmPtr** *dnm*)

frees the DNM

Parameters

dnm	pointer to the DNM to be freed

6.1.3.8 dnmIteratorPtr getDnmChildrenIterator (dnmIteratorPtr *it*)

creates an iterator for the children of the current position of an iterator, i.e. over the sentences of a certain paragraph, or over the words of a sentence. The new iterator has to be free'd manually as well

Parameters

it	the iterator to which tells us what to iterate over

Return values

the	iterator for the children

6.1.3.9 dnmIteratorPtr getDnmIterator (dnmPtr *dnm*, enum dnm_level *level*)

creates an iterator over the document (uses malloc -> has to be free'd later)

Parameters

dnm	the document to be iterated over
level	the chunks we want to iterate over (DNM_LEVEL_PARA, DNM_LEVEL_SENTENCE, DN↩ M_LEVEL_WORD)

Return values

a	pointer to the iterator

6.1.3.10 char∗ getDnmIteratorContent (dnmIteratorPtr *it*)

Returns the plain text of the chunk an iterator points to

Parameters

it	The iterator The plain text, ended by \0, which has to be free'd manually

6.2 llamapun_ngrams.h File Reference

```
#include "jsoninclude.h"
#include <libxml/tree.h>
#include <libxml/parser.h>
#include <libxml/xpath.h>
#include <libxml/xpathInternals.h>
```

Functions

- json_object ∗ llamapun_get_ngrams (xmlDocPtr doc)

6.2.1 Function Documentation

6.2.1.1 json_object∗ llamapun_get_ngrams (xmlDocPtr *doc*)

creates and returns statistics of the unigrams, bigrams, and trigrams found in a document.

Parameters

doc	the DOM

Return values

returns	the counts as a JSON object

6.3 llamapun_para_discr.h File Reference

```
#include "jsoninclude.h"
#include <libxml/tree.h>
```

Functions

- json_object ∗ llamapun_para_discr_get_bags (xmlDocPtr doc)

6.3.1 Function Documentation

6.3.1.1 json_object∗ llamapun_para_discr_get_bags (xmlDocPtr *doc*)

Collects bags of words, i.e. it counts how often which word occurs in which type of paragraph.

- experimental -

  Parameters

doc	The input document with marked up paragraphs

  Return values

Returns	the results as a JSON object.

6.4 stemmer.h File Reference

```
#include <stdio.h>
```

Functions

- void init_stemmer ()
- void close_stemmer ()
- void morpha_stem (const char ∗input, char ∗∗output)

Variables

- FILE ∗ **morpha_instream**
- FILE ∗ **morpha_outstream**
- char ∗ **morpha_instream_buff_ptr**
- char ∗ **morpha_outstream_buff_ptr**

6.4.1 Function Documentation

6.4.1.1 void close_stemmer ( )

closes the stemmer

6.4.1.2 void init_stemmer ( )

Initializes the stemmer

6.4.1.3 void morpha_stem (const char ∗ *input,* char ∗∗ *output*)

stems a sentence

Parameters

input	the input string
output	pointer to the stemmed output

6.5 stopwords.h File Reference

```
#include "jsoninclude.h"
```

Functions

- void read_stopwords_from_json (json_object ∗)
- void load_stopwords ()
- void free_stopwords ()
- int is_stopword (const char ∗word)

6.5.1 Function Documentation

6.5.1.1 void free_stopwords ( )

frees the currently loaded set of stopwords

6.5.1.2 int is_stopword (const char ∗ *word*)

Checks whether a word is regarded a stopword. Note that this function is case sensitive.

Parameters

word	the word to be checked

Return values

returns	1 if the word is a stopword, otherwise 0

6.5.1.3 void load_stopwords ( )

loads the stopwords from a predefined set of math specialized stopwords

6.5.1.4 void read_stopwords_from_json (json_object ∗)

loads the stopwords from a JSON array

6.6 unicode_normalizer.h File Reference

Functions

- void normalize_unicode (char ∗input, char ∗∗output)

6.6.1 Function Documentation

6.6.1.1 void normalize_unicode (char ∗ *input,* char ∗∗ *output*)

Creates a normalized copy of a string

Parameters

input	the input string
output	a pointer to the (normalized) output