Summarizer

Generated by Doxygen 1.8.9.1

Sat Jun 20 2015 18:44:10

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

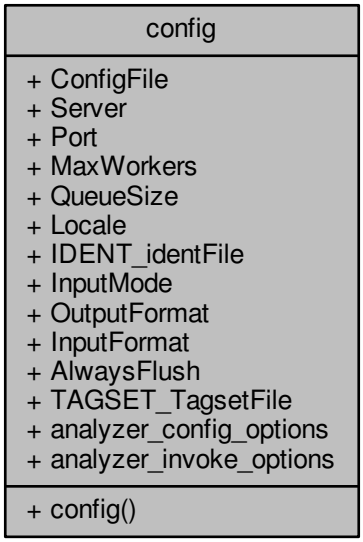
Class Documentation

4.1 config Class Reference

Class config implements a set of specific options for the NLP analyzer, providing a C++ wrapper to libcfg+ library.

`#include <config.h>`

Collaboration diagram for config:



Public Member Functions

- config (int ac, char **av)

  constructor

Public Attributes

- std::string ConfigFile
- bool Server

  Server mode on/off.
- int Port

  port number for server mode
- int MaxWorkers

  Maximum number of workers to fork (i.e. number of simultaneously atended clients)
- int QueueSize

  Size of socket queue (number of clients waiting to be atended without being rejected)
- std::wstring Locale

  Locale of text to process.
- std::wstring IDENT_identFile

  Configuration file for language identifier.
- InputModes InputMode

  Mode used to process input: DOC: load a document, then process it.
- OutputFormats OutputFormat

  Selected input and output format.
- InputFormats InputFormat
- bool AlwaysFlush

  whether splitter buffer must be flushed at each line
- std::wstring TAGSET_TagsetFile

  Tagset to use for shortening tags in output.
- analyzer::config_options analyzer_config_options
- analyzer::invoke_options analyzer_invoke_options

4.1.1 Detailed Description

Class config implements a set of specific options for the NLP analyzer, providing a C++ wrapper to libcfg+ library.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 config::config (int *ac*, char ** *av*) [inline]

constructor

4.1.3 Member Data Documentation

4.1.3.1 bool config::AlwaysFlush

whether splitter buffer must be flushed at each line

4.1.3.2 analyzer::config_options config::analyzer_config_options

4.1.3.3 analyzer::invoke_options config::analyzer_invoke_options

4.1.3.4 std::string config::ConfigFile

4.1.3.5 std::wstring config::IDENT_identFile

Configuration file for language identifier.

4.1.3.6 InputFormats config::InputFormat

4.1.3.7 InputModes config::InputMode

Mode used to process input: DOC: load a document, then process it.

CORPUS: infinite sentence-by-sentnce processing

4.1.3.8 std::wstring config::Locale

Locale of text to process.

4.1.3.9 int config::MaxWorkers

Maximum number of workers to fork (i.e. number of simultaneously atended clients)

4.1.3.10 OutputFormats config::OutputFormat

Selected input and output format.

4.1.3.11 int config::Port

port number for server mode

4.1.3.12 int config::QueueSize

Size of socket queue (number of clients waiting to be atended without being rejected)

4.1.3.13 bool config::Server

Server mode on/off.

4.1.3.14 std::wstring config::TAGSET_TagsetFile

Tagset to use for shortening tags in output.

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

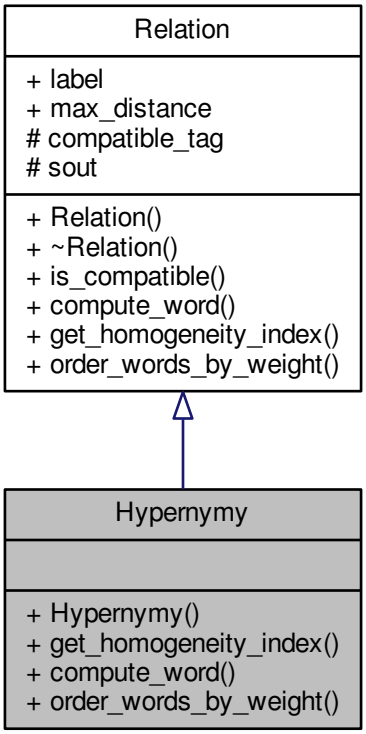
- /home/samuel/Summarizer/src/config.h

4.2 Hypernymy Class Reference

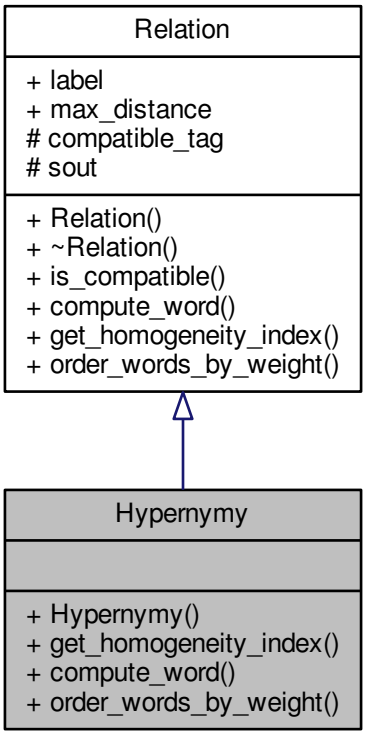
Class Hypernymy represents the hypernymy relation: two words are related if one is an hypernymy of the other and the hypernymy depth is smaller or equal than a given maximum.

```
#include <Relation.h>
```

Inheritance diagram for Hypernymy:



Collaboration diagram for Hypernymy:



Public Member Functions

- Hypernymy (int k, double alpha, const std::wstring &semfile, std::wostream &sout)

  Constructor.
- double get_homogeneity_index (const std::list< word_pos > &words, const std::list< related_words > &relations, const std::unordered_map< std::wstring, std::pair< int, word_pos * > > &unique_words)

  Computes the homogeinity index of the given structures using the specific formula of this relation.
- bool compute_word (const freeling::word &w, const freeling::sentence &s, const freeling::document &doc, int n_paragraph, int n_sentence, int position, std::list< word_pos > &words, std::list< related_words > &relations, std::unordered_map< std::wstring, std::pair< int, word_pos * > > &unique_words) const

  Returns true and stores the word w in the list words, list relations and unordered_map unique_words if w is compatible with the words in these structures using this relation.
- std::list< word_pos > order_words_by_weight (const std::unordered_map< std::wstring, std::pair< int, word_pos * > > &unique_words) const

  Sorts the words in unique_words by word frequency and returns a list with them.

Additional Inherited Members

4.2.1 Detailed Description

Class Hypernymy represents the hypernymy relation: two words are related if one is an hypernymy of the other and the hypernymy depth is smaller or equal than a given maximum.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Hypernymy::Hypernymy (int *k*, double *alpha*, const std::wstring & *semfile*, std::wostream & *sout*)

Constructor.

4.2.3 Member Function Documentation

4.2.3.1 bool Hypernymy::compute_word (const freeling::word & *w*, const freeling::sentence & *s*, const freeling::document & *doc*, int *n_paragraph*, int *n_sentence*, int *position*, std::list< word_pos > & *words*, std::list< related_words > & *relations*, std::unordered_map< std::wstring, std::pair< int, word_pos * > > & *unique_words*) const [virtual]

Returns true and stores the word w in the list words, list relations and unordered_map unique_words if w is compatible with the words in these structures using this relation.

Implements Relation.

4.2.3.2 double Hypernymy::get_homogeneity_index (const std::list< word_pos > & *words*, const std::list< related_words > & *relations*, const std::unordered_map< std::wstring, std::pair< int, word_pos * > > & *unique_words*) [virtual]

Computes the homogeinity index of the given structures using the specific formula of this relation.

Implements Relation.

4.2.3.3 list< word_pos > Hypernymy::order_words_by_weight (const std::unordered_map< std::wstring, std::pair< int, word_pos * > > & *unique_words*) const [virtual]

Sorts the words in unique_words by word frequency and returns a list with them.

Implements Relation.

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

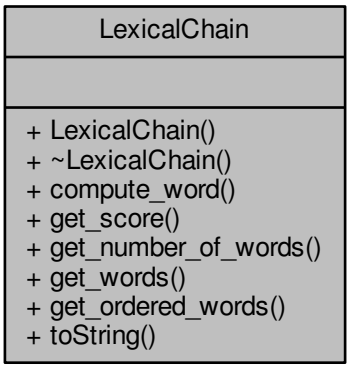
- /home/samuel/Summarizer/src/Relation.h
- /home/samuel/Summarizer/src/Relation.cc

4.3 LexicalChain Class Reference

Class LexicalChain represents a lexical chain and computes words and stores (or not) them into the structures.

`#include <LexicalChain.h>`

Collaboration diagram for LexicalChain:



Public Member Functions

- LexicalChain (Relation *r, const freeling::word &w, const freeling::sentence &s, int n_paragraph, int n_sentence, int position)

  Constructor.
- ~LexicalChain ()

  Destructor.
- bool compute_word (const freeling::word &w, const freeling::sentence &s, const freeling::document &doc, int n_paragraph, int n_sentence, int position, std::wostream &sout)

  Computes a word, if the word an be added to the lexical chain, this method stores it in its structures and return true.
- double get_score ()

  Get the score of the lexical chain.
- int get_number_of_words () const

  Get the number of words inside the lexical chain.
- const std::list< word_pos > & get_words () const

  Get all the words embedded in a word_pos struct of the lexical chain.
- std::list< word_pos > get_ordered_words () const

  Get all the words ordered by frequency.
- std::wstring toString ()

  Get a string representation of the lexical chain to debug.

4.3.1 Detailed Description

Class LexicalChain represents a lexical chain and computes words and stores (or not) them into the structures.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 LexicalChain::LexicalChain (Relation * *r*, const freeling::word & *w*, const freeling::sentence & *s*, int *n_paragraph*, int *n_sentence*, int *position*)

Constructor.

4.3.2.2 LexicalChain::~LexicalChain ( )

Destructor.

4.3.3 Member Function Documentation

4.3.3.1 bool LexicalChain::compute_word (const freeling::word & *w,* const freeling::sentence & *s,* const freeling::document & *doc,* int *n_paragraph,* int *n_sentence,* int *position,* std::wostream & *sout*)

Computes a word, if the word an be added to the lexical chain, this method stores it in its structures and return true. Otherwise, it does nothing and returns false.

4.3.3.2 int LexicalChain::get_number_of_words ( ) const

Get the number of words inside the lexical chain.

4.3.3.3 list< word_pos > LexicalChain::get_ordered_words ( ) const

Get all the words ordered by frequency.

4.3.3.4 double LexicalChain::get_score ( )

Get the score of the lexical chain.

4.3.3.5 const list< word_pos > & LexicalChain::get_words ( ) const

Get all the words embedded in a word_pos struct of the lexical chain.

4.3.3.6 wstring LexicalChain::toString ( )

Get a string representation of the lexical chain to debug.

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

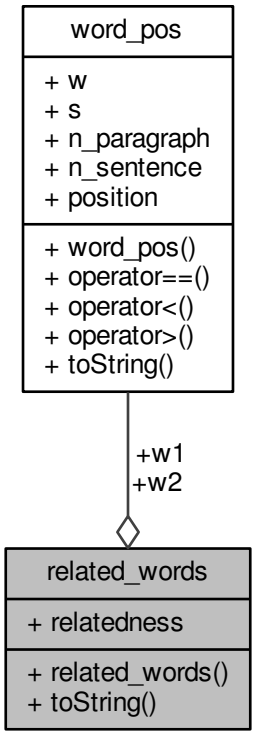
- /home/samuel/Summarizer/src/LexicalChain.h
- /home/samuel/Summarizer/src/LexicalChain.cc

4.4 related_words Struct Reference

Struct that represents a relationship between two words.

`#include <Relation.h>`

Collaboration diagram for related_words:



Public Member Functions

- related_words (const word_pos &w_p1, const word_pos &w_p2, double relatedness)
- std::wstring toString () const

  Get a string representation of the relationship to debug.

Public Attributes

- const word_pos & w1
- const word_pos & w2
- double relatedness

  Relatedness represents the strength of the relationship.

4.4.1 Detailed Description

Struct that represents a relationship between two words.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 related_words::related_words (const word_pos & *w_p1,* const word_pos & *w_p2,* double *relatedness*)

4.4.3 Member Function Documentation

4.4.3.1 wstring related_words::toString ( ) const

Get a string representation of the relationship to debug.

4.4.4 Member Data Documentation

4.4.4.1 double related_words::relatedness

Relatedness represents the strength of the relationship.

4.4.4.2 const word_pos& related_words::w1

4.4.4.3 const word_pos& related_words::w2

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

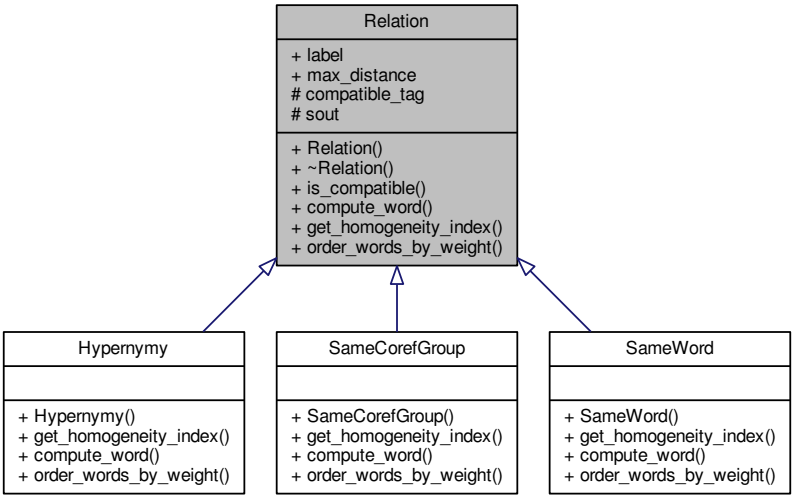
- /home/samuel/Summarizer/src/Relation.h
- /home/samuel/Summarizer/src/Relation.cc

4.5 Relation Class Reference

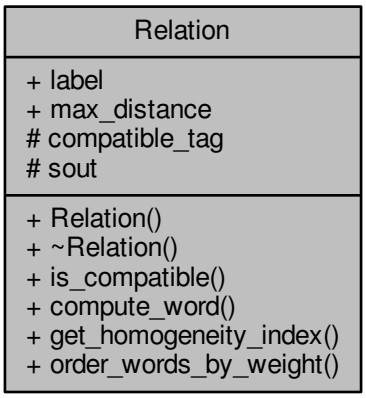
Class Relation is a non-instantiable class which defines many virtual methods to check if a word is compatible with the Relation or if a word can be stored in the structures of a lexical chain.

`#include <Relation.h>`

Inheritance diagram for Relation:



Collaboration diagram for Relation:



Public Member Functions

- Relation (const std::wstring s, const std::wstring t)

  Constructor.
- ∼Relation ()

  Destructor.
- bool is_compatible (const freeling::word &w) const

  True if the words tag is compatible with the relation.
- virtual bool compute_word (const freeling::word &w, const freeling::sentence &s, const freeling::document &doc, int n_paragraph, int n_sentence, int position, std::list< word_pos > &words, std::list< related_words > &relations, std::unordered_map< std::wstring, std::pair< int, word_pos ∗ > > &unique_words) const =0
- virtual double get_homogeneity_index (const std::list< word_pos > &words, const std::list< related_words > &relations, const std::unordered_map< std::wstring, std::pair< int, word_pos ∗ > > &unique_words)=0
- virtual std::list< word_pos > order_words_by_weight (const std::unordered_map< std::wstring, std::pair< int, word_pos ∗ > > &unique_words) const =0

Public Attributes

- const std::wstring label

  Label with the name of the related. It is used for debugging.

Static Public Attributes

- static int max_distance = 0

  The maximum distance in phrases between two words to be related.

Protected Attributes

- const freeling::regexp compatible_tag

  If a word tag matchs with compatible_tag, then the word is compatible with the relation.

- std::wostream * sout
  Pointer to a wostream to debug.

4.5.1 Detailed Description

Class Relation is a non-instantiable class which defines many virtual methods to check if a word is compatible with the Relation or if a word can be stored in the structures of a lexical chain.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Relation::Relation (const std::wstring *s*, const std::wstring *t*)

Constructor.

4.5.2.2 Relation::~Relation ( )

Destructor.

4.5.3 Member Function Documentation

4.5.3.1 bool Relation::compute_word (const freeling::word & *w*, const freeling::sentence & *s*, const freeling::document & *doc*, int *n_paragraph*, int *n_sentence*, int *position*, std::list< word_pos > & *words*, std::list< related_words > & *relations*, std::unordered_map< std::wstring, std::pair< int, word_pos * > > & *unique_words*) const [pure virtual]

Implemented in SameCorefGroup, Hypernymy, and SameWord.

4.5.3.2 virtual double Relation::get_homogeneity_index (const std::list< word_pos > & *words*, const std::list< related_words > & *relations*, const std::unordered_map< std::wstring, std::pair< int, word_pos * > > & *unique_words*) [pure virtual]

Implemented in SameCorefGroup, Hypernymy, and SameWord.

4.5.3.3 bool Relation::is_compatible (const freeling::word & *w*) const

True if the words tag is compatible with the relation.

4.5.3.4 virtual std::list<word_pos> Relation::order_words_by_weight (const std::unordered_map< std::wstring, std::pair< int, word_pos * > > & *unique_words*) const [pure virtual]

Implemented in SameCorefGroup, Hypernymy, and SameWord.

4.5.4 Member Data Documentation

4.5.4.1 const freeling::regexp Relation::compatible_tag [protected]

If a word tag matchs with compatible_tag, then the word is compatible with the relation.

4.5.4.2 const std::wstring Relation::label

Label with the name of the related. It is used for debugging.

4.5.4.3 int Relation::max_distance = 0 [static]

The maximum distance in phrases between two words to be related.

4.5.4.4 std::wostream* Relation::sout [protected]

Pointer to a wostream to debug.

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

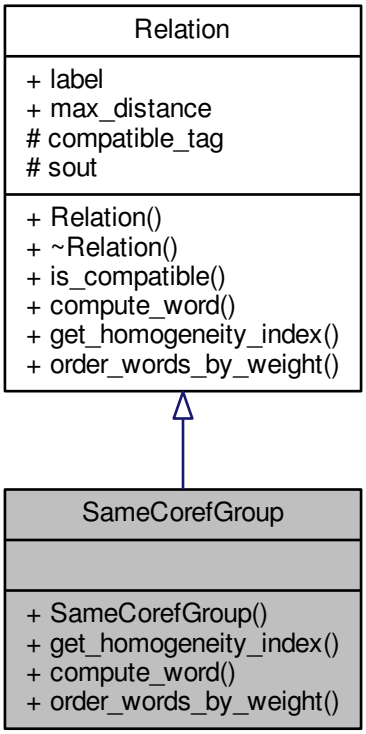
- /home/samuel/Summarizer/src/Relation.h
- /home/samuel/Summarizer/src/Relation.cc

4.6 SameCorefGroup Class Reference

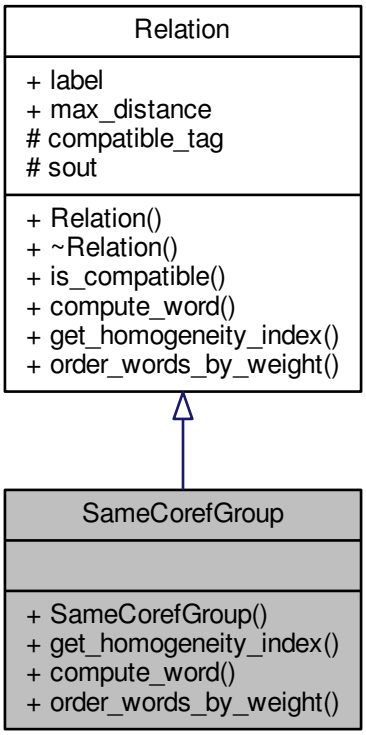
Class SameCorefGroup represents the same coreference group relation: two words are related if they are in the same coreference group.

`#include <Relation.h>`

Inheritance diagram for SameCorefGroup:



Collaboration diagram for SameCorefGroup:



Public Member Functions

- SameCorefGroup (std::wostream &sout)

  Constructor.
- double get_homogeneity_index (const std::list< word_pos > &words, const std::list< related_words > &relations, const std::unordered_map< std::wstring, std::pair< int, word_pos * > > &unique_words)

  Computes the homogeinity index of the given structures using the specific formula of this relation.
- bool compute_word (const freeling::word &w, const freeling::sentence &s, const freeling::document &doc, int n_paragraph, int n_sentence, int position, std::list< word_pos > &words, std::list< related_words > &relations, std::unordered_map< std::wstring, std::pair< int, word_pos * > > &unique_words) const

  Returns true and stores the word w in the list words, list relations and unordered_map unique_words if w is compatible with the words in these structures using this relation.
- std::list< word_pos > order_words_by_weight (const std::unordered_map< std::wstring, std::pair< int, word_pos * > > &unique_words) const

  Sorts the words in unique_words by word frequency and returns a list with them.

Additional Inherited Members

4.6.1 Detailed Description

Class SameCorefGroup represents the same coreference group relation: two words are related if they are in the same coreference group.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 SameCorefGroup::SameCorefGroup (std::wostream & *sout*)

Constructor.

4.6.3 Member Function Documentation

4.6.3.1 bool SameCorefGroup::compute_word (const freeling::word & *w,* const freeling::sentence & *s,* const freeling::document & *doc,* int *n_paragraph,* int *n_sentence,* int *position,* std::list< word_pos > & *words,* std::list< related_words > & *relations,* std::unordered_map< std::wstring, std::pair< int, word_pos * > > & *unique_words*) const [virtual]

Returns true and stores the word w in the list words, list relations and unordered_map unique_words if w is compatible with the words in these structures using this relation.

Implements Relation.

4.6.3.2 double SameCorefGroup::get_homogeneity_index (const std::list< word_pos > & *words,* const std::list< related_words > & *relations,* const std::unordered_map< std::wstring, std::pair< int, word_pos * > > & *unique_words*) [virtual]

Computes the homogeinity index of the given structures using the specific formula of this relation.

Implements Relation.

4.6.3.3 list< word_pos > SameCorefGroup::order_words_by_weight (const std::unordered_map< std::wstring, std::pair< int, word_pos * > > & *unique_words*) const [virtual]

Sorts the words in unique_words by word frequency and returns a list with them.

Implements Relation.

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

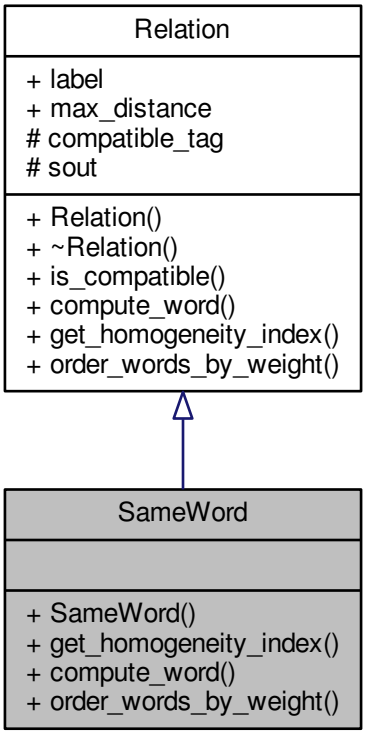
- /home/samuel/Summarizer/src/Relation.h
- /home/samuel/Summarizer/src/Relation.cc

4.7 SameWord Class Reference

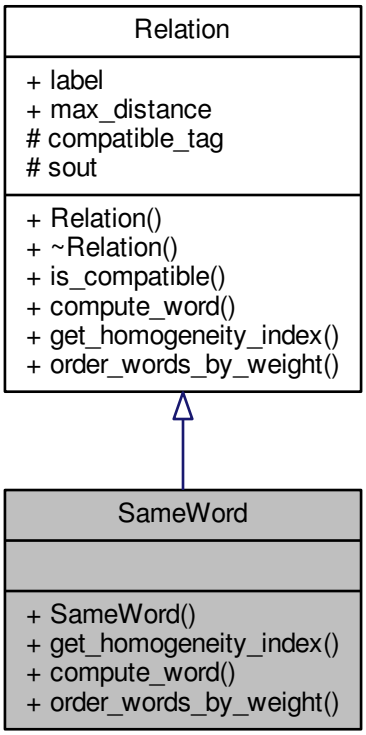
Class SameWord represents the same word relation: two words are related if they are the same word.

`#include <Relation.h>`

Inheritance diagram for SameWord:



Collaboration diagram for SameWord:



Public Member Functions

- SameWord (std::wostream &sout)

  Constructor.
- double get_homogeneity_index (const std::list< word_pos > &words, const std::list< related_words > &relations, const std::unordered_map< std::wstring, std::pair< int, word_pos * > > &unique_words)

  Computes the homogeinity index of the given structures using the specific formula of this relation.
- bool compute_word (const freeling::word &w, const freeling::sentence &s, const freeling::document &doc, int n_paragraph, int n_sentence, int position, std::list< word_pos > &words, std::list< related_words > &relations, std::unordered_map< std::wstring, std::pair< int, word_pos * > > &unique_words) const

  Returns true and stores the word w in the list words, list relations and unordered_map unique_words if w is compatible with the words in these structures using this relation.
- std::list< word_pos > order_words_by_weight (const std::unordered_map< std::wstring, std::pair< int, word_pos * > > &unique_words) const

  In SameWord, the words in unique_words are not sorted because there is just one word.

Additional Inherited Members

4.7.1 Detailed Description

Class SameWord represents the same word relation: two words are related if they are the same word.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 SameWord::SameWord (std::wostream & *sout*)

Constructor.

4.7.3 Member Function Documentation

4.7.3.1 bool SameWord::compute_word (const freeling::word & *w,* const freeling::sentence & *s,* const freeling::document & *doc,* int *n_paragraph,* int *n_sentence,* int *position,* std::list< word_pos > & *words,* std::list< related_words > & *relations,* std::unordered_map< std::wstring, std::pair< int, word_pos * > > & *unique_words*) const [virtual]

Returns true and stores the word w in the list words, list relations and unordered_map unique_words if w is compatible with the words in these structures using this relation.

Implements Relation.

4.7.3.2 double SameWord::get_homogeneity_index (const std::list< word_pos > & *words,* const std::list< related_words > & *relations,* const std::unordered_map< std::wstring, std::pair< int, word_pos * > > & *unique_words*) [virtual]

Computes the homogeinity index of the given structures using the specific formula of this relation.

Implements Relation.

4.7.3.3 list< word_pos > SameWord::order_words_by_weight (const std::unordered_map< std::wstring, std::pair< int, word_pos * > > & *unique_words*) const [virtual]

In SameWord, the words in unique_words are not sorted because there is just one word.

It returns the word_pos in unique_words in a list.

Implements Relation.

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

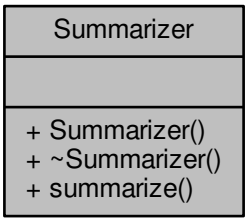
- /home/samuel/Summarizer/src/Relation.h
- /home/samuel/Summarizer/src/Relation.cc

4.8 Summarizer Class Reference

Summarizer class summarizes a document using the lexical chains method.

`#include <Summarizer.h>`

Collaboration diagram for Summarizer:



Public Member Functions

- Summarizer (const std::wstring &datFile, bool debug)

  Constructor.
- ∼Summarizer ()

  Destructor.
- std::list< const freeling::sentence * > summarize (std::wostream &sout, const freeling::document &doc)

  Summarizes a document and returns the list of sentences that composes the summary.

4.8.1 Detailed Description

Summarizer class summarizes a document using the lexical chains method.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Summarizer::Summarizer (const std::wstring & *datFile,* bool *debug*)

Constructor.

4.8.2.2 Summarizer::∼Summarizer ( )

Destructor.

4.8.3 Member Function Documentation

4.8.3.1 list< const sentence * > Summarizer::summarize (std::wostream & *sout,* const freeling::document & *doc*)

Summarizes a document and returns the list of sentences that composes the summary.

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

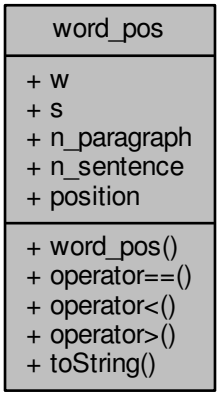
- /home/samuel/Summarizer/src/Summarizer.h
- /home/samuel/Summarizer/src/Summarizer.cc

4.9 word_pos Struct Reference

Struct that allow us to compare words easily.

`#include <Relation.h>`

Collaboration diagram for word_pos:



Public Member Functions

- word_pos (const freeling::word &w_p, const freeling::sentence &s_p, int n_paragraph, int n_sentence, int position)
- bool operator== (word_pos other) const

  Two words are equal if they are in the same position, phrase and paragraph.
- bool operator< (word_pos other) const

  One word is smaller than other one if it appears later in the text.
- bool operator> (word_pos other) const

  One word is greater than other one if it appears sooner in the text.
- std::wstring toString () const

  Get a string representation of the word_pos to debug.

Public Attributes

- const freeling::word & w
- const freeling::sentence & s
- int n_paragraph
- int n_sentence
- int position

4.9.1 Detailed Description

Struct that allow us to compare words easily.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 word_pos::word_pos (const freeling::word & *w_p,* const freeling::sentence & *s_p,* int *n_paragraph,* int *n_sentence,* int *position*)

4.9.3 Member Function Documentation

4.9.3.1 bool word_pos::operator< (word_pos *other*) const

One word is smaller than other one if it appears later in the text.

4.9.3.2 bool word_pos::operator== (word_pos *other*) const

Two words are equal if they are in the same position, phrase and paragraph.

4.9.3.3 bool word_pos::operator> (word_pos *other*) const

One word is greater than other one if it appears sooner in the text.

4.9.3.4 wstring word_pos::toString ( ) const

Get a string representation of the word_pos to debug.

4.9.4 Member Data Documentation

4.9.4.1 int word_pos::n_paragraph

4.9.4.2 int word_pos::n_sentence

4.9.4.3 int word_pos::position

4.9.4.4 const freeling::sentence& word_pos::s

4.9.4.5 const freeling::word& word_pos::w

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

- /home/samuel/Summarizer/src/Relation.h
- /home/samuel/Summarizer/src/Relation.cc

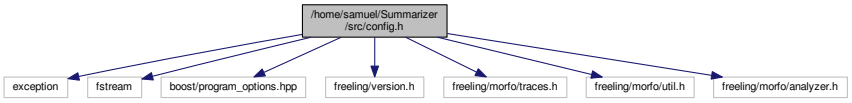
Chapter 5

File Documentation

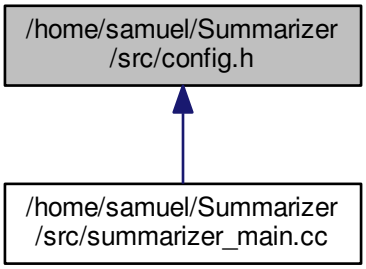
5.1 /home/samuel/Summarizer/src/config.h File Reference

```
#include <exception>
#include <fstream>
#include <boost/program_options.hpp>
#include "freeling/version.h"
#include "freeling/morfo/traces.h"
#include "freeling/morfo/util.h"
#include "freeling/morfo/analyzer.h"
```

Include dependency graph for config.h:



This graph shows which files directly or indirectly include this file:



Classes

- class config

Class config implements a set of specific options for the NLP analyzer, providing a C++ wrapper to libcfg+ library.

Macros

- #define MOD_TRACENAME L"CONFIG_OPTIONS"
- #define DEFAULT_MAX_WORKERS 5
- #define DEFAULT_QUEUE_SIZE 32

Enumerations

- enum InputModes { MODE_CORPUS, MODE_DOC }
- enum OutputFormats {
  OUT_FREELING, OUT_TRAIN, OUT_CONLL, OUT_XML,
  OUT_JSON, OUT_NAF }
- enum InputFormats { INP_TEXT, INP_FREELING, INP_CONLL }

5.1.1 Macro Definition Documentation

5.1.1.1 #define DEFAULT_MAX_WORKERS 5

5.1.1.2 #define DEFAULT_QUEUE_SIZE 32

5.1.1.3 #define MOD_TRACENAME L"CONFIG_OPTIONS"

5.1.2 Enumeration Type Documentation

5.1.2.1 enum InputFormats

Enumerator

INP_TEXT

INP_FREELING

INP_CONLL

5.1.2.2 enum InputModes

Enumerator

MODE_CORPUS

MODE_DOC

5.1.2.3 enum OutputFormats

Enumerator

OUT_FREELING

OUT_TRAIN

OUT_CONLL

OUT_XML

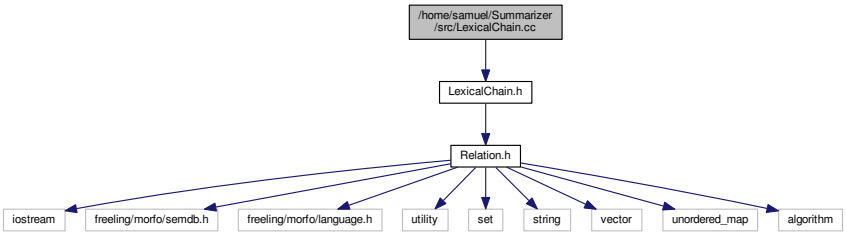
OUT_JSON

OUT_NAF

5.2 /home/samuel/Summarizer/src/LexicalChain.cc File Reference

```
#include "LexicalChain.h"
```

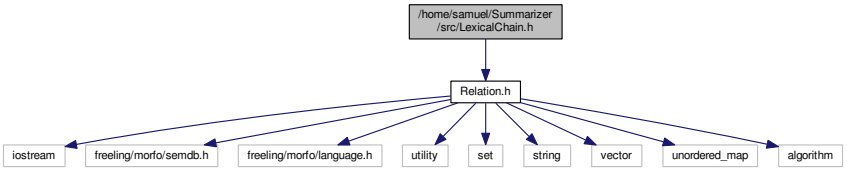
Include dependency graph for LexicalChain.cc:



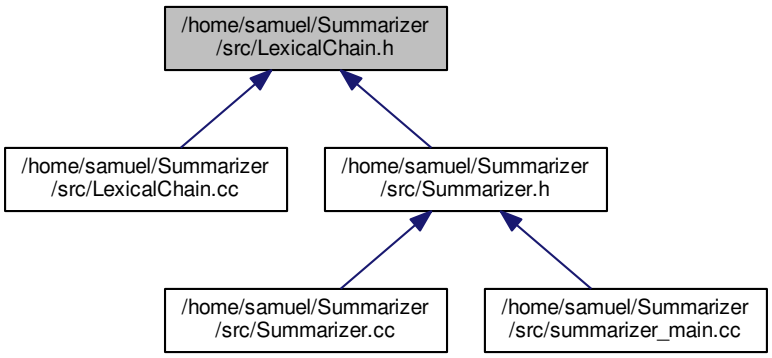
5.3 /home/samuel/Summarizer/src/LexicalChain.h File Reference

```
#include "Relation.h"
```

Include dependency graph for LexicalChain.h:



This graph shows which files directly or indirectly include this file:



Classes

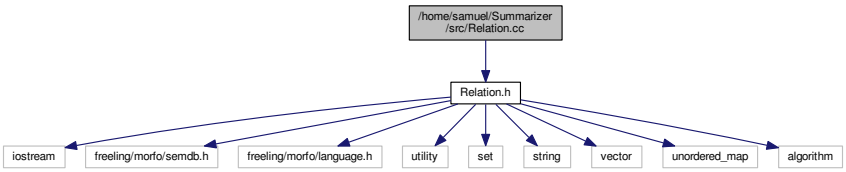
- class LexicalChain

Class LexicalChain represents a lexical chain and computes words and stores (or not) them into the structures.

5.4 /home/samuel/Summarizer/src/Relation.cc File Reference

```
#include "Relation.h"
```

Include dependency graph for Relation.cc:



Functions

- bool order_by_score (const pair< int, word_pos * > &p1, const pair< int, word_pos * > &p2)
- bool order_by_tag_and_score (const pair< int, word_pos * > &p1, const pair< int, word_pos * > &p2)

5.4.1 Function Documentation

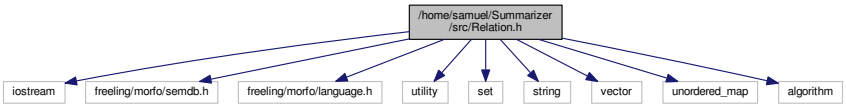
5.4.1.1 bool order_by_score (const pair< int, word_pos * > & *p1,* const pair< int, word_pos * > & *p2*)

5.4.1.2 bool order_by_tag_and_score (const pair< int, word_pos * > & *p1,* const pair< int, word_pos * > & *p2*)

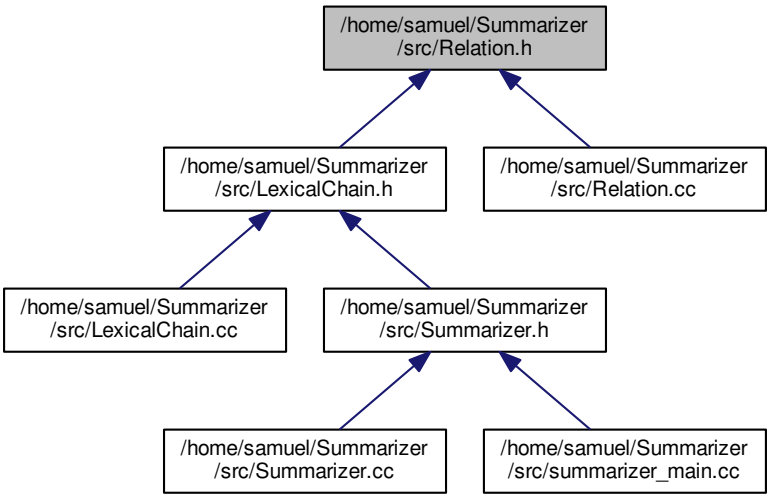
5.5 /home/samuel/Summarizer/src/Relation.h File Reference

```
#include <iostream>
#include "freeling/morfo/semdb.h"
#include "freeling/morfo/language.h"
#include <utility>
#include <set>
#include <string>
#include <vector>
#include <unordered_map>
#include <algorithm>
```

Include dependency graph for Relation.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct word_pos

  Struct that allow us to compare words easily.
- struct related_words

  Struct that represents a relationship between two words.
- class Relation

  Class Relation is a non-instantiable class which defines many virtual methods to check if a word is compatible with the Relation or if a word can be stored in the structures of a lexical chain.
- class SameWord

  Class SameWord represents the same word relation: two words are related if they are the same word.
- class Hypernymy

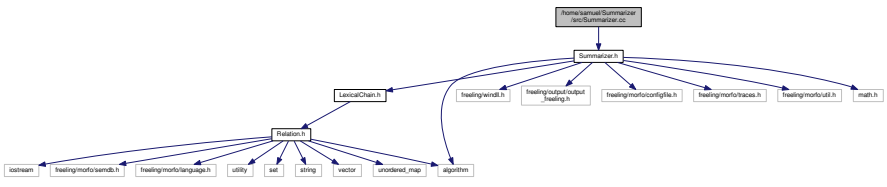
  Class Hypernymy represents the hypernymy relation: two words are related if one is an hypernymy of the other and the hypernymy depth is smaller or equal than a given maximum.
- class SameCorefGroup

  Class SameCorefGroup represents the same coreference group relation: two words are related if they are in the same coreference group.

5.6 /home/samuel/Summarizer/src/Summarizer.cc File Reference

```
#include "Summarizer.h"
```

Include dependency graph for Summarizer.cc:



Functions

- bool compare_lexical_chains (LexicalChain &first, LexicalChain &second)
- bool order_by_scores (const pair< int, const word_pos * > &sc_wp1, const pair< int, const word_pos * > &sc_wp2)

5.6.1 Function Documentation

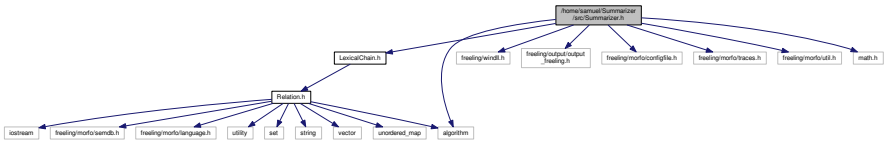
5.6.1.1 bool compare_lexical_chains (LexicalChain & *first,* LexicalChain & *second*)

5.6.1.2 bool order_by_scores (const pair< int, const word_pos * > & *sc_wp1,* const pair< int, const word_pos * > & *sc_wp2*)

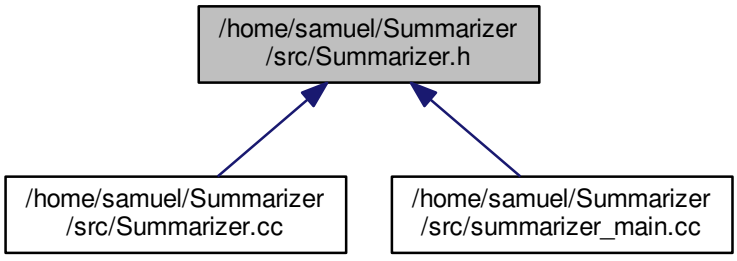
5.7 /home/samuel/Summarizer/src/Summarizer.h File Reference

```
#include "LexicalChain.h"
#include "freeling/windll.h"
#include "freeling/output/output_freeling.h"
#include "freeling/morfo/configfile.h"
#include "freeling/morfo/traces.h"
#include "freeling/morfo/util.h"
#include <algorithm>
#include <math.h>
```

Include dependency graph for Summarizer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Summarizer

  Summarizer class summarizes a document using the lexical chains method.

Macros

- #define MOD_TRACENAME L"ANALYZER"

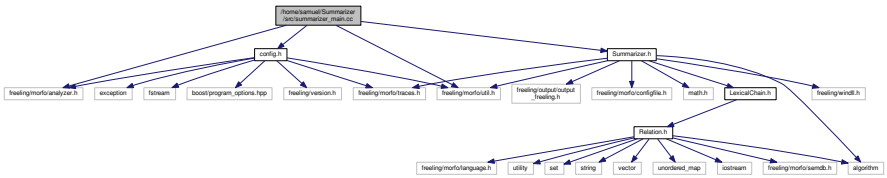
5.7.1 Macro Definition Documentation

5.7.1.1 #define MOD_TRACENAME L"ANALYZER"

5.8 /home/samuel/Summarizer/src/summarizer_main.cc File Reference

```
#include "freeling/morfo/analyzer.h"
#include "freeling/morfo/util.h"
#include "config.h"
#include "Summarizer.h"
```

Include dependency graph for summarizer_main.cc:



Functions

- analyzer::config_options fill_config (const wstring &path)

  predeclarations
- analyzer::invoke_options fill_invoke ()

  Load an ad-hoc set of invoke options.
- int main (int argc, char **argv)

5.8.1 Function Documentation

5.8.1.1 analyzer::config_options fill_config (const wstring & *path*)

predeclarations

Load an ad-hoc set of configuration options. Language of text to process

Tokenizer configuration file

Splitter configuration file

Morphological analyzer options

NEC config file

Sense annotator and WSD config files

Tagger options

Chart parser config file

Dependency parsers config files

Coreference resolution config file

5.8.1.2 analyzer::invoke_options fill_invoke ( )

Load an ad-hoc set of invoke options.

Level of analysis in input and output

activate/deactivate morphological analyzer modules

5.8.1.3 int main (int *argc,* char ∗∗ *argv*)

read FreeLing installation path if given, use default otherwise

set config options (which modules to create, with which configuration)

create analyzer

set invoke options (which modules to use)

load invoke options into analyzer

load document to analyze

analyze text, leave result in doc

create summarizer

summarize document

print the summary

Index