

## Summarizer

Generated by Doxygen 1.8.9.1

Sat Jun 20 2015 18:44:10



# Contents

<b>1</b>	<b>Hierarchical Index</b>	<b>1</b>
1.1	Class Hierarchy	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List	5
<b>4</b>	<b>Class Documentation</b>	<b>7</b>
4.1	config Class Reference	7
4.1.1	Detailed Description	8
4.1.2	Constructor & Destructor Documentation	8
4.1.2.1	config	8
4.1.3	Member Data Documentation	8
4.1.3.1	AlwaysFlush	8
4.1.3.2	analyzer_config_options	8
4.1.3.3	analyzer_invoke_options	8
4.1.3.4	ConfigFile	8
4.1.3.5	IDENT_identFile	8
4.1.3.6	InputFormat	9
4.1.3.7	InputMode	9
4.1.3.8	Locale	9
4.1.3.9	MaxWorkers	9
4.1.3.10	OutputFormat	9
4.1.3.11	Port	9
4.1.3.12	QueueSize	9
4.1.3.13	Server	9
4.1.3.14	TAGSET_TagsetFile	9
4.2	Hypernymy Class Reference	9
4.2.1	Detailed Description	11
4.2.2	Constructor & Destructor Documentation	12

4.2.2.1	Hypernymy . . . . .	12
4.2.3	Member Function Documentation . . . . .	12
4.2.3.1	compute_word . . . . .	12
4.2.3.2	get_homogeneity_index . . . . .	12
4.2.3.3	order_words_by_weight . . . . .	12
4.3	LexicalChain Class Reference . . . . .	12
4.3.1	Detailed Description . . . . .	13
4.3.2	Constructor & Destructor Documentation . . . . .	13
4.3.2.1	LexicalChain . . . . .	13
4.3.2.2	~LexicalChain . . . . .	14
4.3.3	Member Function Documentation . . . . .	14
4.3.3.1	compute_word . . . . .	14
4.3.3.2	get_number_of_words . . . . .	14
4.3.3.3	get_ordered_words . . . . .	14
4.3.3.4	get_score . . . . .	14
4.3.3.5	get_words . . . . .	14
4.3.3.6	toString . . . . .	14
4.4	related_words Struct Reference . . . . .	14
4.4.1	Detailed Description . . . . .	15
4.4.2	Constructor & Destructor Documentation . . . . .	15
4.4.2.1	related_words . . . . .	15
4.4.3	Member Function Documentation . . . . .	16
4.4.3.1	toString . . . . .	16
4.4.4	Member Data Documentation . . . . .	16
4.4.4.1	relatedness . . . . .	16
4.4.4.2	w1 . . . . .	16
4.4.4.3	w2 . . . . .	16
4.5	Relation Class Reference . . . . .	16
4.5.1	Detailed Description . . . . .	18
4.5.2	Constructor & Destructor Documentation . . . . .	18
4.5.2.1	Relation . . . . .	18
4.5.2.2	~Relation . . . . .	18
4.5.3	Member Function Documentation . . . . .	18
4.5.3.1	compute_word . . . . .	18
4.5.3.2	get_homogeneity_index . . . . .	18
4.5.3.3	is_compatible . . . . .	18
4.5.3.4	order_words_by_weight . . . . .	18
4.5.4	Member Data Documentation . . . . .	18
4.5.4.1	compatible_tag . . . . .	18
4.5.4.2	label . . . . .	18

4.5.4.3	max_distance	19
4.5.4.4	sout	19
4.6	SameCorefGroup Class Reference	19
4.6.1	Detailed Description	20
4.6.2	Constructor & Destructor Documentation	21
4.6.2.1	SameCorefGroup	21
4.6.3	Member Function Documentation	21
4.6.3.1	compute_word	21
4.6.3.2	get_homogeneity_index	21
4.6.3.3	order_words_by_weight	21
4.7	SameWord Class Reference	21
4.7.1	Detailed Description	23
4.7.2	Constructor & Destructor Documentation	24
4.7.2.1	SameWord	24
4.7.3	Member Function Documentation	24
4.7.3.1	compute_word	24
4.7.3.2	get_homogeneity_index	24
4.7.3.3	order_words_by_weight	24
4.8	Summarizer Class Reference	24
4.8.1	Detailed Description	25
4.8.2	Constructor & Destructor Documentation	25
4.8.2.1	Summarizer	25
4.8.2.2	~Summarizer	25
4.8.3	Member Function Documentation	25
4.8.3.1	summarize	25
4.9	word_pos Struct Reference	26
4.9.1	Detailed Description	26
4.9.2	Constructor & Destructor Documentation	27
4.9.2.1	word_pos	27
4.9.3	Member Function Documentation	27
4.9.3.1	operator<	27
4.9.3.2	operator==	27
4.9.3.3	operator>	27
4.9.3.4	toString	27
4.9.4	Member Data Documentation	27
4.9.4.1	n_paragraph	27
4.9.4.2	n_sentence	27
4.9.4.3	position	27
4.9.4.4	s	27
4.9.4.5	w	27

<b>5 File Documentation</b>	<b>29</b>
5.1 /home/samuel/Summarizer/src/config.h File Reference	29
5.1.1 Macro Definition Documentation	30
5.1.1.1 DEFAULT_MAX_WORKERS	30
5.1.1.2 DEFAULT_QUEUE_SIZE	30
5.1.1.3 MOD_TRACENAME	30
5.1.2 Enumeration Type Documentation	30
5.1.2.1 InputFormats	30
5.1.2.2 InputModes	30
5.1.2.3 OutputFormats	30
5.2 /home/samuel/Summarizer/src/LexicalChain.cc File Reference	31
5.3 /home/samuel/Summarizer/src/LexicalChain.h File Reference	31
5.4 /home/samuel/Summarizer/src/Relation.cc File Reference	32
5.4.1 Function Documentation	32
5.4.1.1 order_by_score	32
5.4.1.2 order_by_tag_and_score	32
5.5 /home/samuel/Summarizer/src/Relation.h File Reference	32
5.6 /home/samuel/Summarizer/src/Summarizer.cc File Reference	33
5.6.1 Function Documentation	34
5.6.1.1 compare_lexical_chains	34
5.6.1.2 order_by_scores	34
5.7 /home/samuel/Summarizer/src/Summarizer.h File Reference	34
5.7.1 Macro Definition Documentation	35
5.7.1.1 MOD_TRACENAME	35
5.8 /home/samuel/Summarizer/src/summarizer_main.cc File Reference	35
5.8.1 Function Documentation	35
5.8.1.1 fill_config	35
5.8.1.2 fill_invoke	36
5.8.1.3 main	36
<b>Index</b>	<b>37</b>

# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

config . . . . .	7
LexicalChain . . . . .	12
related_words . . . . .	14
Relation . . . . .	16
Hypernymy . . . . .	9
SameCorefGroup . . . . .	19
SameWord . . . . .	21
Summarizer . . . . .	24
word_pos . . . . .	26





## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">config</a>	Class <a href="#">config</a> implements a set of specific options for the NLP analyzer, providing a C++ wrapper to libcfg+ library . . . . .	7
<a href="#">Hypernymy</a>	Class <a href="#">Hypernymy</a> 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 . . . . .	9
<a href="#">LexicalChain</a>	Class <a href="#">LexicalChain</a> represents a lexical chain and computes words and stores (or not) them into the structures . . . . .	12
<a href="#">related_words</a>	Struct that represents a relationship between two words . . . . .	14
<a href="#">Relation</a>	Class <a href="#">Relation</a> is a non-instantiable class which defines many virtual methods to check if a word is compatible with the <a href="#">Relation</a> or if a word can be stored in the structures of a lexical chain . . . . .	16
<a href="#">SameCorefGroup</a>	Class <a href="#">SameCorefGroup</a> represents the same coreference group relation: two words are related if they are in the same coreference group . . . . .	19
<a href="#">SameWord</a>	Class <a href="#">SameWord</a> represents the same word relation: two words are related if they are the same word . . . . .	21
<a href="#">Summarizer</a>	<a href="#">Summarizer</a> class summarizes a document using the lexical chains method . . . . .	24
<a href="#">word_pos</a>	Struct that allow us to compare words easily . . . . .	26



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

/home/samuel/Summarizer/src/ <a href="#">config.h</a> . . . . .	29
/home/samuel/Summarizer/src/ <a href="#">LexicalChain.cc</a> . . . . .	31
/home/samuel/Summarizer/src/ <a href="#">LexicalChain.h</a> . . . . .	31
/home/samuel/Summarizer/src/ <a href="#">Relation.cc</a> . . . . .	32
/home/samuel/Summarizer/src/ <a href="#">Relation.h</a> . . . . .	32
/home/samuel/Summarizer/src/ <a href="#">Summarizer.cc</a> . . . . .	33
/home/samuel/Summarizer/src/ <a href="#">Summarizer.h</a> . . . . .	34
/home/samuel/Summarizer/src/ <a href="#">summarizer_main.cc</a> . . . . .	35



## 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:

config
<div>+ ConfigFile</div> <div>+ Server</div> <div>+ Port</div> <div>+ MaxWorkers</div> <div>+ QueueSize</div> <div>+ Locale</div> <div>+ IDENT_identFile</div> <div>+ InputMode</div> <div>+ OutputFormat</div> <div>+ InputFormat</div> <div>+ AlwaysFlush</div> <div>+ TAGSET_TagsetFile</div> <div>+ analyzer_config_options</div> <div>+ analyzer_invoke_options</div>
<div>+ config()</div>

#### 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 attended clients)*
- `int` [QueueSize](#)  
*Size of socket queue (number of clients waiting to be attended 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-sentence 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 attended 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 attended 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:

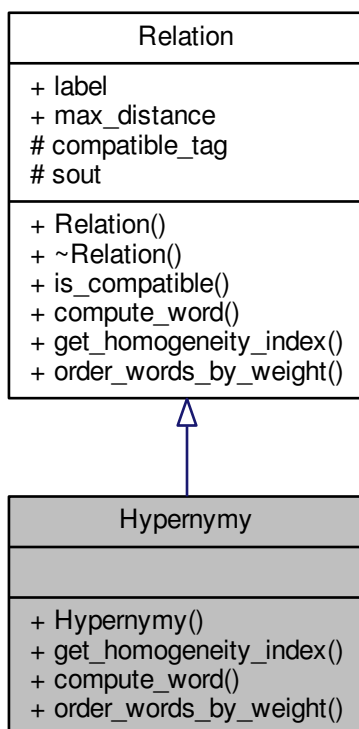
- /home/samuel/Summarizer/src/[config.h](#)

## 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.

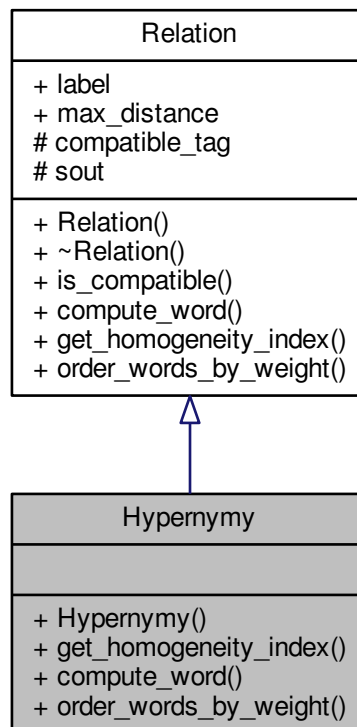
```
#include <Relation.h>
```

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 homogeneity index of the given structures using the specific formula of this relation.*
- bool [compute\\_word](#) (const freeing::word &w, const freeing::sentence &s, const freeing::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 homogeneity 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:

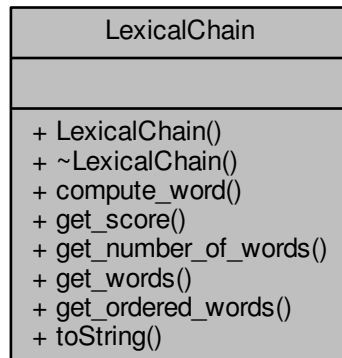
- `/home/samuel/Summarizer/src/Relation.h`
- `/home/samuel/Summarizer/src/Relation.cc`

## 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 can 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 can 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:

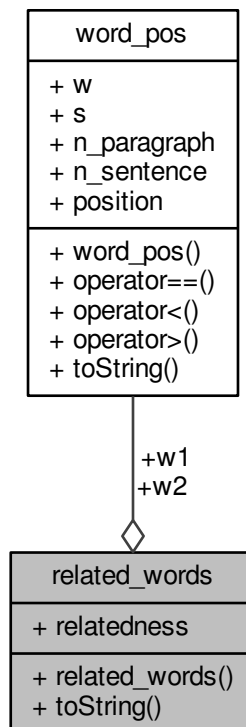
- /home/samuel/Summarizer/src/[LexicalChain.h](#)
- /home/samuel/Summarizer/src/[LexicalChain.cc](#)

## 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:

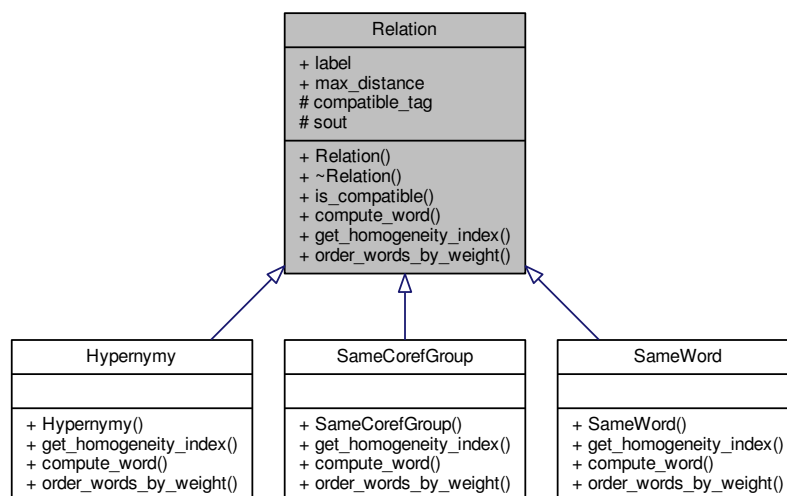
- </home/samuel/Summarizer/src/Relation.h>
- </home/samuel/Summarizer/src/Relation.cc>

## 4.5 Relation Class Reference

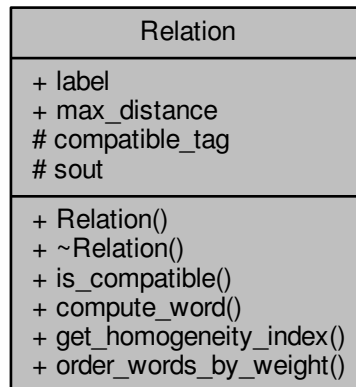
Class [Relation](#) is a non-instantiable class which defines many virtual methods to check if a word is compatible with the [Relation](#) or if a word can be stored in the structures of a lexical chain.

```
#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 freeing::word &w) const  
*True if the words tag is compatible with the relation.*
- virtual bool [compute\\_word](#) (const freeing::word &w, const freeing::sentence &s, const freeing::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 freeing::regexp [compatible\\_tag](#)  
*If a word 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 matches 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:

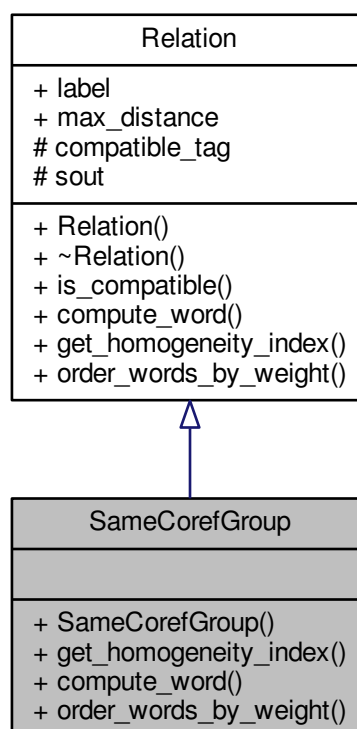
- </home/samuel/Summarizer/src/Relation.h>
- </home/samuel/Summarizer/src/Relation.cc>

## 4.6 SameCorefGroup Class Reference

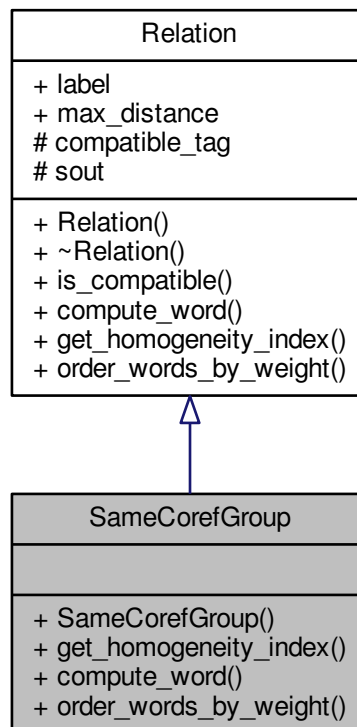
Class [SameCorefGroup](#) represents the same coreference group relation: two words are related if they are in the same coreference group.

```
#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 homogeneity index of the given structures using the specific formula of this relation.*
- bool [compute\\_word](#) (`const freeing::word &w`, `const freeing::sentence &s`, `const freeing::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 freeing::word & w, const freeing::sentence & s, const freeing::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 homogeneity 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:

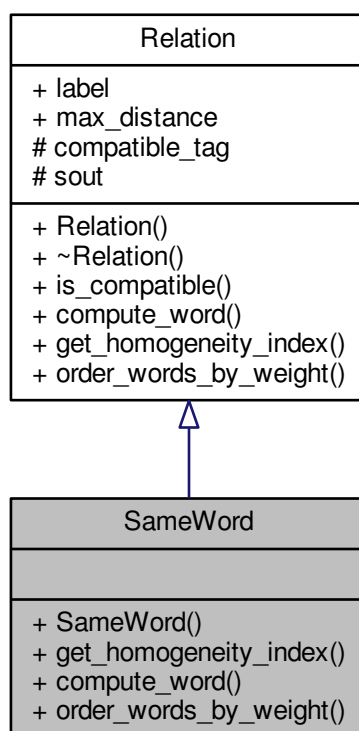
- `/home/samuel/Summarizer/src/Relation.h`
- `/home/samuel/Summarizer/src/Relation.cc`

## 4.7 SameWord Class Reference

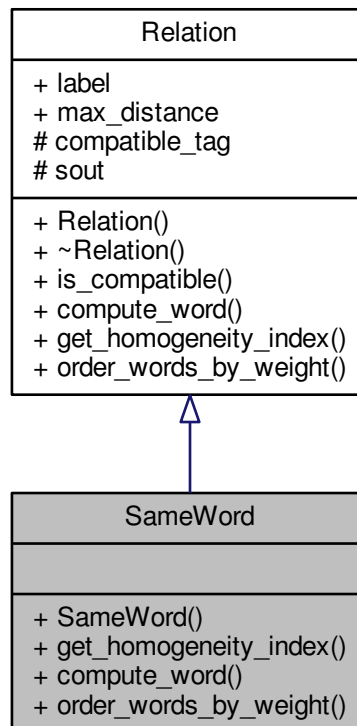
Class [SameWord](#) represents the same word relation: two words are related if they are the same word.

```
#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 homogeneity index of the given structures using the specific formula of this relation.*

- bool [compute\\_word](#) (const freeing::word &w, const freeing::sentence &s, const freeing::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 freeing::word & w, const freeing::sentence & s, const freeing::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 homogeneity 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:

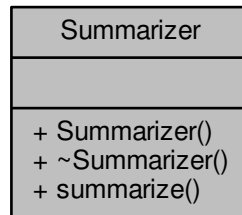
- [/home/samuel/Summarizer/src/Relation.h](#)
- [/home/samuel/Summarizer/src/Relation.cc](#)

## 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:

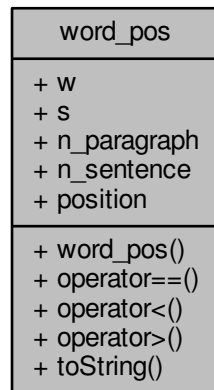
- /home/samuel/Summarizer/src/[Summarizer.h](#)
- /home/samuel/Summarizer/src/[Summarizer.cc](#)

## 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 freeing::word &w\_p, const freeing::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 freeing::word & [w](#)
- const freeing::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 freeing::word & w_p, const freeing::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 freeing::sentence& word_pos::s`

4.9.4.5 `const freeing::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](#)



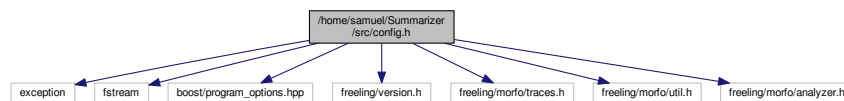
## Chapter 5

# File Documentation

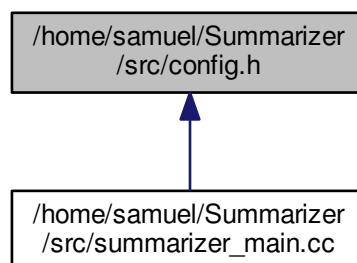
### 5.1 /home/samuel/Summarizer/src/config.h File Reference

```
#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`

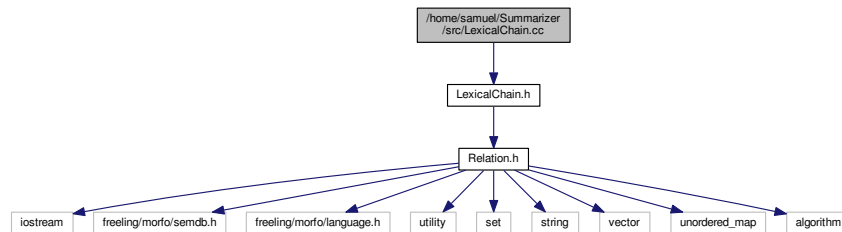
Enumerator

**`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"
```

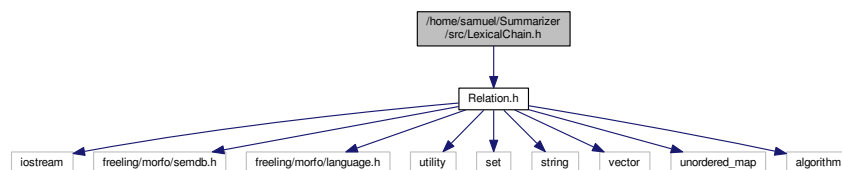
Include dependency graph for LexicalChain.cc:



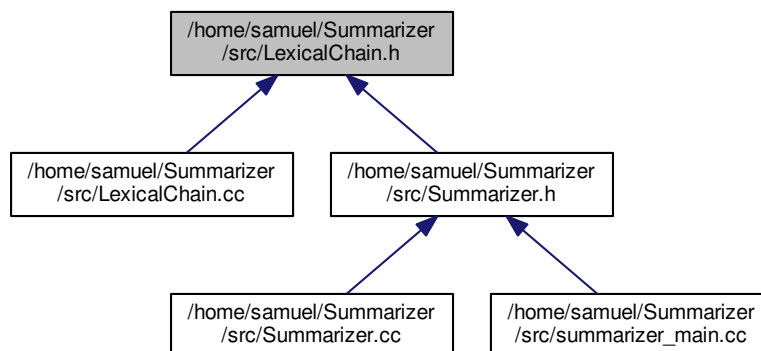
## 5.3 /home/samuel/Summarizer/src/LexicalChain.h File Reference

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

Include dependency graph for LexicalChain.h:



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



### Classes

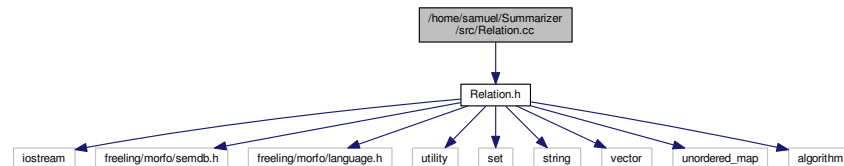
- class [LexicalChain](#)

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

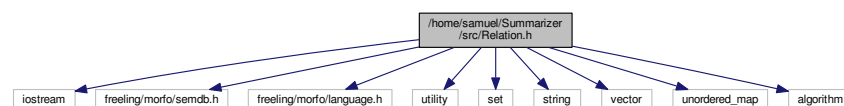
5.4.1.1 bool [order\\_by\\_score](#) ( const pair< int, [word\\_pos](#) \* > &p1, const pair< int, [word\\_pos](#) \* > &p2 )

5.4.1.2 bool [order\\_by\\_tag\\_and\\_score](#) ( const pair< int, [word\\_pos](#) \* > &p1, const pair< int, [word\\_pos](#) \* > &p2 )

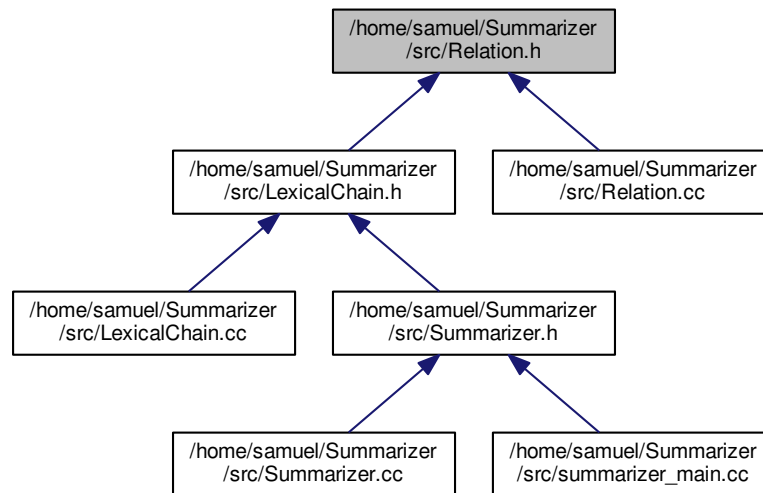
## 5.5 /home/samuel/Summarizer/src/Relation.h File Reference

```
#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:



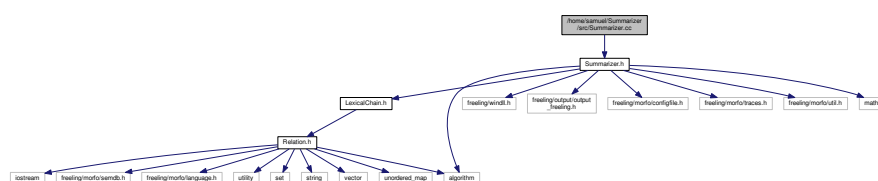
## Classes

- 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

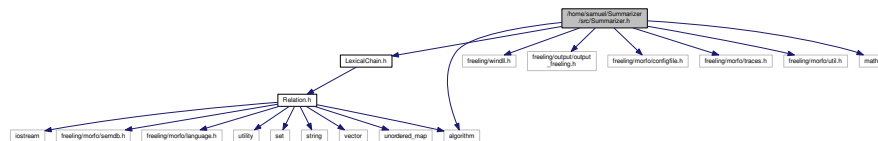
5.6.1.1 bool [compare\\_lexical\\_chains](#) ( [LexicalChain](#) & *first*, [LexicalChain](#) & *second* )

5.6.1.2 bool [order\\_by\\_scores](#) ( const pair< int, const [word\\_pos](#) \* > & *sc\_wp1*, const pair< int, const [word\\_pos](#) \* > & *sc\_wp2* )

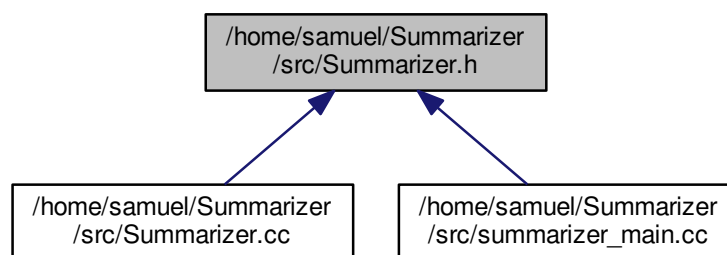
## 5.7 /home/samuel/Summarizer/src/Summarizer.h File Reference

```
#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

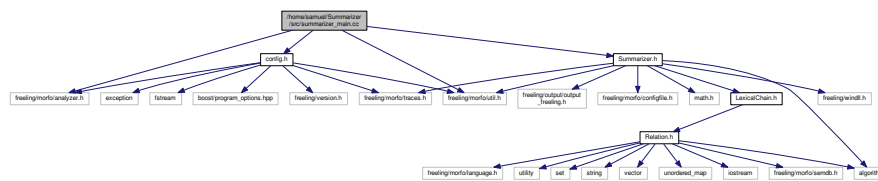
- #define MOD\_TRACENAME L"ANALYZER"

### 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

/home/samuel/Summarizer/src/LexicalChain.cc, [31](#)  
/home/samuel/Summarizer/src/LexicalChain.h, [31](#)  
/home/samuel/Summarizer/src/Relation.cc, [32](#)  
/home/samuel/Summarizer/src/Relation.h, [32](#)  
/home/samuel/Summarizer/src/Summarizer.cc, [33](#)  
/home/samuel/Summarizer/src/Summarizer.h, [34](#)  
/home/samuel/Summarizer/src/config.h, [29](#)  
/home/samuel/Summarizer/src/summarizer\_main.cc, [35](#)  
~LexicalChain  
    LexicalChain, [13](#)  
~Relation  
    Relation, [18](#)  
~Summarizer  
    Summarizer, [25](#)  
  
AlwaysFlush  
    config, [8](#)  
analyzer\_config\_options  
    config, [8](#)  
analyzer\_invoke\_options  
    config, [8](#)  
  
compare\_lexical\_chains  
    Summarizer.cc, [34](#)  
compatible\_tag  
    Relation, [18](#)  
compute\_word  
    Hypernymy, [12](#)  
    LexicalChain, [14](#)  
    Relation, [18](#)  
    SameCorefGroup, [21](#)  
    SameWord, [24](#)  
config, [7](#)  
    AlwaysFlush, [8](#)  
    analyzer\_config\_options, [8](#)  
    analyzer\_invoke\_options, [8](#)  
    config, [8](#)  
    ConfigFile, [8](#)  
    IDENT\_identFile, [8](#)  
    InputFormat, [8](#)  
    InputMode, [9](#)  
    Locale, [9](#)  
    MaxWorkers, [9](#)  
    OutputFormat, [9](#)  
    Port, [9](#)  
    QueueSize, [9](#)  
    Server, [9](#)  
    TAGSET\_TagsetFile, [9](#)  
config.h  
    DEFAULT\_MAX\_WORKERS, [30](#)  
    DEFAULT\_QUEUE\_SIZE, [30](#)  
    INP\_CONLL, [30](#)  
    INP\_FREELING, [30](#)  
    INP\_TEXT, [30](#)  
    InputFormats, [30](#)  
    InputModes, [30](#)  
    MOD\_TRACENAME, [30](#)  
    MODE\_CORPUS, [30](#)  
    MODE\_DOC, [30](#)  
    OUT\_CONLL, [30](#)  
    OUT\_FREELING, [30](#)  
    OUT\_JSON, [30](#)  
    OUT\_NAF, [30](#)  
    OUT\_TRAIN, [30](#)  
    OUT\_XML, [30](#)  
    OutputFormats, [30](#)  
ConfigFile  
    config, [8](#)  
  
DEFAULT\_MAX\_WORKERS  
    config.h, [30](#)  
DEFAULT\_QUEUE\_SIZE  
    config.h, [30](#)  
  
fill\_config  
    summarizer\_main.cc, [35](#)  
fill\_invoke  
    summarizer\_main.cc, [35](#)  
  
get\_homogeneity\_index  
    Hypernymy, [12](#)  
    Relation, [18](#)  
    SameCorefGroup, [21](#)  
    SameWord, [24](#)  
get\_number\_of\_words  
    LexicalChain, [14](#)  
get\_ordered\_words  
    LexicalChain, [14](#)  
get\_score  
    LexicalChain, [14](#)  
get\_words  
    LexicalChain, [14](#)  
  
Hypernymy, [9](#)  
    compute\_word, [12](#)  
    get\_homogeneity\_index, [12](#)  
    Hypernymy, [12](#)  
    order\_words\_by\_weight, [12](#)  
  
IDENT\_identFile  
    config, [8](#)

- INP\_CONLL
  - config.h, 30
- INP\_FREELING
  - config.h, 30
- INP\_TEXT
  - config.h, 30
- InputFormat
  - config, 8
- InputFormats
  - config.h, 30
- InputMode
  - config, 9
- InputModes
  - config.h, 30
- is\_compatible
  - Relation, 18
- label
  - Relation, 18
- LexicalChain, 12
  - ~LexicalChain, 13
  - compute\_word, 14
  - get\_number\_of\_words, 14
  - get\_ordered\_words, 14
  - get\_score, 14
  - get\_words, 14
  - LexicalChain, 13
  - toString, 14
- Locale
  - config, 9
- MOD\_TRACENAME
  - config.h, 30
  - Summarizer.h, 35
- MODE\_CORPUS
  - config.h, 30
- MODE\_DOC
  - config.h, 30
- main
  - summarizer\_main.cc, 36
- max\_distance
  - Relation, 18
- MaxWorkers
  - config, 9
- n\_paragraph
  - word\_pos, 27
- n\_sentence
  - word\_pos, 27
- OUT\_CONLL
  - config.h, 30
- OUT\_FREELING
  - config.h, 30
- OUT\_JSON
  - config.h, 30
- OUT\_NAF
  - config.h, 30
- OUT\_TRAIN
  - config.h, 30
- OUT\_XML
  - config.h, 30
- operator<
  - word\_pos, 27
- operator>
  - word\_pos, 27
- operator==
  - word\_pos, 27
- order\_by\_score
  - Relation.cc, 32
- order\_by\_scores
  - Summarizer.cc, 34
- order\_by\_tag\_and\_score
  - Relation.cc, 32
- order\_words\_by\_weight
  - Hypernymy, 12
  - Relation, 18
  - SameCorefGroup, 21
  - SameWord, 24
- OutputFormat
  - config, 9
- OutputFormats
  - config.h, 30
- Port
  - config, 9
- position
  - word\_pos, 27
- QueueSize
  - config, 9
- related\_words, 14
  - related\_words, 15
  - relatedness, 16
  - toString, 16
  - w1, 16
  - w2, 16
- relatedness
  - related\_words, 16
- Relation, 16
  - ~Relation, 18
  - compatible\_tag, 18
  - compute\_word, 18
  - get\_homogeneity\_index, 18
  - is\_compatible, 18
  - label, 18
  - max\_distance, 18
  - order\_words\_by\_weight, 18
  - Relation, 18
  - sout, 19
- Relation.cc
  - order\_by\_score, 32
  - order\_by\_tag\_and\_score, 32
- s
  - word\_pos, 27
- SameCorefGroup, 19

- compute\_word, [21](#)
- get\_homogeneity\_index, [21](#)
- order\_words\_by\_weight, [21](#)
- SameCorefGroup, [21](#)
- SameWord, [21](#)
  - compute\_word, [24](#)
  - get\_homogeneity\_index, [24](#)
  - order\_words\_by\_weight, [24](#)
  - SameWord, [24](#)
- Server
  - config, [9](#)
- sout
  - Relation, [19](#)
- summarize
  - Summarizer, [25](#)
- Summarizer, [24](#)
  - ~Summarizer, [25](#)
  - summarize, [25](#)
  - Summarizer, [25](#)
- Summarizer.cc
  - compare\_lexical\_chains, [34](#)
  - order\_by\_scores, [34](#)
- Summarizer.h
  - MOD\_TRACENAME, [35](#)
- summarizer\_main.cc
  - fill\_config, [35](#)
  - fill\_invoke, [35](#)
  - main, [36](#)
- TAGSET\_TagsetFile
  - config, [9](#)
- toString
  - LexicalChain, [14](#)
  - related\_words, [16](#)
  - word\_pos, [27](#)
- w
  - word\_pos, [27](#)
- w1
  - related\_words, [16](#)
- w2
  - related\_words, [16](#)
- word\_pos, [26](#)
  - n\_paragraph, [27](#)
  - n\_sentence, [27](#)
  - operator<, [27](#)
  - operator>, [27](#)
  - operator==, [27](#)
  - position, [27](#)
  - s, [27](#)
  - toString, [27](#)
  - w, [27](#)
  - word\_pos, [27](#)