

Calculate sentence score

1.0

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>File Index</b>	<b>1</b>
1.1	File List . . . . .	1
<b>2</b>	<b>File Documentation</b>	<b>3</b>
2.1	src/functions.cpp File Reference . . . . .	3
2.2	src/functions.hpp File Reference . . . . .	3
2.2.1	Function Documentation . . . . .	4
2.2.1.1	contains_duplicates() . . . . .	4
2.2.1.2	contains_duplicates_h() . . . . .	5
2.2.1.3	correct_dict_hashes_and_extract_duplicates() . . . . .	5
2.2.1.4	correct_text_hashes() . . . . .	6
2.2.1.5	extract_words_count() . . . . .	7
2.2.1.6	hash_function() . . . . .	7
2.2.1.7	init_hashes() . . . . .	8
2.2.1.8	init_logging() . . . . .	9
2.2.1.9	init_random_matrices() [1/2] . . . . .	9
2.2.1.10	init_random_matrices() [2/2] . . . . .	10
2.2.1.11	print_bitset() . . . . .	10
2.2.1.12	print_bitset_vector() . . . . .	11
2.2.1.13	print_duplicates() . . . . .	11
2.2.1.14	print_hash_table() . . . . .	11
2.2.1.15	print_matrix() . . . . .	12
2.2.1.16	print_vector() . . . . .	12
2.2.1.17	print_words() . . . . .	12

2.2.1.18	<a href="#">read_dict_from_file()</a>	12
2.2.1.19	<a href="#">read_text_from_file()</a>	13
2.2.1.20	<a href="#">search_and_calculate_matrices()</a>	14
2.2.1.21	<a href="#">write_in_file()</a>	15
2.3	<a href="#">src/main.cpp File Reference</a>	16
2.3.1	<a href="#">Macro Definition Documentation</a>	16
2.3.1.1	<a href="#">matrix_size</a>	16
2.3.1.2	<a href="#">max_dict_size</a>	16
2.3.1.3	<a href="#">max_stop_words_size</a>	17
2.3.1.4	<a href="#">max_term_size</a>	17
2.3.2	<a href="#">Function Documentation</a>	17
2.3.2.1	<a href="#">main()</a>	17
2.4	<a href="#">src/unit_tests.cpp File Reference</a>	18
2.4.1	<a href="#">Macro Definition Documentation</a>	19
2.4.1.1	<a href="#">BOOST_TEST_MODULE</a>	20
2.4.1.2	<a href="#">matrix_size</a>	20
2.4.1.3	<a href="#">max_dict_size</a>	20
2.4.1.4	<a href="#">max_stop_words_size</a>	20
2.4.1.5	<a href="#">max_term_size</a>	20
2.4.2	<a href="#">Function Documentation</a>	20
2.4.2.1	<a href="#">BOOST_AUTO_TEST_CASE()</a> [1/7]	21
2.4.2.2	<a href="#">BOOST_AUTO_TEST_CASE()</a> [2/7]	21
2.4.2.3	<a href="#">BOOST_AUTO_TEST_CASE()</a> [3/7]	22
2.4.2.4	<a href="#">BOOST_AUTO_TEST_CASE()</a> [4/7]	22
2.4.2.5	<a href="#">BOOST_AUTO_TEST_CASE()</a> [5/7]	23
2.4.2.6	<a href="#">BOOST_AUTO_TEST_CASE()</a> [6/7]	23
2.4.2.7	<a href="#">BOOST_AUTO_TEST_CASE()</a> [7/7]	24
2.4.2.8	<a href="#">dict_hashes()</a>	24
2.4.2.9	<a href="#">dictionary()</a>	24
2.4.2.10	<a href="#">hash_table()</a>	24

2.4.2.11	<code>matrices()</code>	25
2.4.2.12	<code>output()</code>	25
2.4.2.13	<code>stop_words()</code>	25
2.4.2.14	<code>stop_words_hashes()</code>	25
2.4.2.15	<code>text()</code>	25
2.4.2.16	<code>text_double_term_hashes()</code>	26
2.4.2.17	<code>text_single_term_hashes()</code>	26
2.4.2.18	<code>text_triple_term_hashes()</code>	26
2.4.3	Variable Documentation	26
2.4.3.1	<code>dict_size</code>	26
2.4.3.2	<code>dictionary_filename</code>	26
2.4.3.3	<code>duplicates</code>	27
2.4.3.4	<code>duplicates_size</code>	27
2.4.3.5	<code>hasher</code>	27
2.4.3.6	<code>stop_words_filename</code>	27
2.4.3.7	<code>stop_words_size</code>	27
2.4.3.8	<code>text_filename</code>	28
2.4.3.9	<code>text_size</code>	28
2.4.3.10	<code>words_count</code>	28



# Chapter 1

## File Index

### 1.1 File List

Here is a list of all files with brief descriptions:

<a href="#">src/functions.cpp</a>	3
<a href="#">src/functions.hpp</a>	3
<a href="#">src/main.cpp</a>	16
<a href="#">src/unit_tests.cpp</a>	18





## Chapter 2

# File Documentation

### 2.1 src/functions.cpp File Reference

```
#include "functions.hpp"
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <time.h>
#include <algorithm>
#include <functional>
#include <map>
#include <set>
#include <random>
#include <string.h>
#include <chrono>
#include <boost/log/core.hpp>
#include <boost/log/trivial.hpp>
#include <boost/log/expressions.hpp>
#include <boost/log/utility/setup/file.hpp>
#include <boost/log/utility/setup/common_attributes.hpp>
```

Include dependency graph for functions.cpp:

### 2.2 src/functions.hpp File Reference

```
#include <vector>
#include <string>
#include <bitset>
#include <map>
```

Include dependency graph for functions.hpp: This graph shows which files directly or indirectly include this file:

#### Functions

- void [init\\_logging](#) ()  
*This function helps in logging level.*
- void [init\\_random\\_matrices](#) (std::vector< std::bitset< 100 >> &[matrices](#))  
*initializing random matrices*

- void `read_dict_from_file` (const std::string &dictionary\_filename, std::vector< std::string > &dictionary)  
*reading dictionary from file*
- void `read_text_from_file` (const std::string &text\_filename, std::vector< std::string > &text)  
*reading input text, stop words from file*
- void `init_hashes` (std::vector< size\_t > &hashes, const std::hash< std::string > &hasher, const std::vector< std::string > &words)  
*calculating hashes of words stored in words*
- void `correct_dict_hashes_and_extract_duplicates` (std::vector< std::vector< size\_t >> &duplicates, std::vector< size\_t > &hashes)  
*finding duplicates and correcting dictionary hashes*
- void `correct_text_hashes` (std::vector< size\_t > &text\_hashes, const std::vector< std::string > &text, const std::vector< std::vector< size\_t >> &duplicates, const std::vector< std::string > &dictionary)  
*correcting input text hashes, maybe there are duplicates*
- void `hash_function` (std::vector< std::pair< size\_t, size\_t >> &hash\_table, const std::vector< size\_t > &hashes)  
*hashing open adresssing with linear probing algorithm*
- void `search_and_calculate_matrices` (std::bitset< 100 > &output, std::map< std::string, size\_t > &words\_count, const std::vector< size\_t > &text\_single\_term\_hashes, const std::vector< size\_t > &text\_double\_term\_hashes, const std::vector< size\_t > &text\_triple\_term\_hashes, const std::vector< size\_t > &stop\_words\_hashes, const std::vector< std::pair< size\_t, size\_t >> &hash\_table, const std::vector< std::bitset< 100 >> &matrices, const std::vector< std::string > &dictionary)  
*correcting input text hashes, maybe there are duplicates*
- void `extract_words_count` (const std::map< std::string, size\_t > &m)
- void `write_in_file` ()
- void `print_hash_table` (const std::vector< std::pair< size\_t, size\_t >> &hash\_table, size\_t begin=0, size\_t end=466548)
- void `print_duplicates` (const std::vector< std::vector< size\_t >> &duplicates, const std::vector< std::string > &dictionary)
- void `print_matrix` (const std::vector< size\_t > &matrix)
- void `print_words` (const std::vector< std::string > &dictionary)
- void `print_vector` (const std::vector< size\_t > &vector)
- void `print_bitset` (const std::bitset< 100 > &bitset)
- void `print_bitset_vector` (const std::bitset< 100 > &bitset)
- bool `contains_duplicates` (std::vector< size\_t > a)
- bool `contains_duplicates_h` (std::vector< size\_t > a)
- void `init_random_matrices` (std::vector< std::vector< size\_t >> &matrices)

## 2.2.1 Function Documentation

### 2.2.1.1 contains\_duplicates()

```
bool contains_duplicates (
    std::vector< size_t > a )
```

Definition at line 426 of file functions.cpp.

```

427 {
428     if (a.size() < 2)
429     {
430         return false;
431     }
432     sort(a.begin(), a.end());
433     //std::cout << a[a.size() - 1] << std::endl;
434     //std::cout << a[0] << std::endl;
435     for (int i = 0; i < a.size() - 1; i++)
436     {
437         if (a[i] == a[i + 1])
438         {
439             std::cout << a[i] << std::endl;
440         }
441     }
442     return false;
443 }

```

### 2.2.1.2 contains\_duplicates\_h()

```

bool contains_duplicates_h (
    std::vector< size_t > a )

```

Definition at line 445 of file functions.cpp.

```

446 {
447     if (a.size() < 2)
448     {
449         return false;
450     }
451
452     for (int i = 0; i < a.size() - 1; i++)
453     {
454         a[i] = a[i] % a.size();
455     }
456
457     sort(a.begin(), a.end());
458     //std::cout << a[a.size() - 1] << std::endl;
459     //std::cout << a[0] << std::endl;
460     for (int i = 0; i < a.size() - 1; i++)
461     {
462         if (a[i] == a[i + 1])
463         {
464             std::cout << a[i] << std::endl;
465         }
466     }
467     return false;
468 }

```

### 2.2.1.3 correct\_dict\_hashes\_and\_extract\_duplicates()

```

void correct_dict_hashes_and_extract_duplicates (
    std::vector< std::vector< size_t >> & duplicates,
    std::vector< size_t > & hashes )

```

finding duplicates and correcting dictionary hashes

#### Parameters

<i>duplicates</i>	- contain words old hashes, indexes to the words with that hashes, and new hashes
<i>hashes</i>	- hash of dictionary, contain hashed strings of each word in dictionary

Definition at line 171 of file functions.cpp.

References duplicates.

Referenced by BOOST\_AUTO\_TEST\_CASE(), and main().

```

174 {
175     auto start = std::chrono::steady_clock::now();
176     size_t size = hashes.size();
177     std::map<size_t, size_t> m;
178
179     if (size < 2)
180     {
181         return;
182     }
183
184     for (size_t i = 0; i < size; ++i)
185     {
186         m[hashes[i]] = 0;
187     }
188     for (size_t i = 0; i < size; ++i)
189     {
190         if (m[hashes[i]])
191         {
192             size_t temp = hashes[i];
193             while (m[temp])
194             {
195                 temp++;
196             }
197             duplicates.push_back({ hashes[i], i, temp });
198             hashes[i] = temp;
199         }
200         else
201         {
202             m[hashes[i]] = 1;
203         }
204     }
205     auto end = std::chrono::steady_clock::now();
206     BOOST_LOG_TRIVIAL(debug) << "correct_dict_hashes_and_extract_duplicates() "
207     << std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count() << " ms";
208 }

```

Here is the caller graph for this function:

#### 2.2.1.4 correct\_text\_hashes()

```

void correct_text_hashes (
    std::vector< size_t > & text_hashes,
    const std::vector< std::string > & text,
    const std::vector< std::vector< size_t >> & duplicates,
    const std::vector< std::string > & dictionary )

```

correcting input text hashes, maybe there are duplicates

##### Parameters

<i>text_hashes</i>	- hash of text, contain hashed strings of each word in text
<i>text</i>	- contain all words in input text(maximum 500terms)
<i>duplicates</i>	- contain words old hashes, indexes to the words with that hashes, and new hashes
<i>dictionary</i>	- conatin all words in dictionary.txt

Definition at line 210 of file functions.cpp.

References duplicates, and text\_size.

Referenced by main().

```

214 {
215     auto start = std::chrono::steady_clock::now();
216     size_t duplicate_size = duplicates.size();
217     size_t text_size = text_hashes.size();
218     for (size_t i = 0; i < duplicates.size(); ++i)
219     {
220         for (size_t j = 0; j < text_size; ++j)
221         {
222             if (text_hashes[j] == duplicates[i][0])
223             {
224                 if (text[j] == dictionary[duplicates[i][1]])
225                 {
226                     text_hashes[j] = duplicates[i][2];
227                 }
228             }
229         }
230     }
231     auto end = std::chrono::steady_clock::now();
232     BOOST_LOG_TRIVIAL(debug) << "correct_text_hashes() "
233     << std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count() << " ms";
234 }

```

Here is the caller graph for this function:

### 2.2.1.5 extract\_words\_count()

```

void extract_words_count (
    const std::map< std::string, size_t > & m )

```

Definition at line 327 of file functions.cpp.

Referenced by main().

```

328 {
329     typedef std::function<bool(std::pair<std::string, int>, std::pair<std::string, int>)> Comparator;
330
331     Comparator compFuncor = [](std::pair<std::string, size_t> elem1 ,std::pair<std::string, size_t> elem2)
332     {
333         return elem1.second > elem2.second;
334     };
335
336     std::set<std::pair<std::string, int>, Comparator> setOfWords(
337         m.begin(), m.end(), compFuncor);
338
339     // Iterate over a set using range base for loop
340     // It will display the items in sorted order of values
341     for (std::pair<std::string, int> element : setOfWords)
342         BOOST_LOG_TRIVIAL(info) << element.first << " :: " << element.second;
343 }

```

Here is the caller graph for this function:

### 2.2.1.6 hash\_function()

```

void hash_function (
    std::vector< std::pair< size_t, size_t >> & hash_table,
    const std::vector< size_t > & hashes )

```

hashing open addressing with linear probing algorithm

#### Parameters

<i>hash_table</i>	- hash table contain hashes and indexes to the words in dictionary
<i>hashes</i>	- hash of dictionary, contain hashed strings of each word in dictionary

Definition at line 236 of file functions.cpp.

References hash\_table(), and words\_count.

Referenced by BOOST\_AUTO\_TEST\_CASE(), and main().

```

238 {
239     auto start = std::chrono::steady_clock::now();
240     size_t words_count = hashes.size();
241     for (size_t i = 0; i < words_count; ++i)
242     {
243         size_t index = hashes[i] % words_count;
244         while (hash_table[index].first)
245         {
246             index = (index + 1) % words_count;
247         }
248         hash_table[index] = std::make_pair(hashes[i], i);
249     }
250     auto end = std::chrono::steady_clock::now();
251     BOOST_LOG_TRIVIAL(debug) << "hash_function()    "
252         << std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count() << " ms";
253 }
```

Here is the call graph for this function: Here is the caller graph for this function:

### 2.2.1.7 init\_hashes()

```

void init_hashes (
    std::vector< size_t > & hashes,
    const std::hash< std::string > & hasher,
    const std::vector< std::string > & words )
```

calculating hashes of words stored in words

#### Parameters

<i>hasher</i>	- object that hashes words
<i>words</i>	- all words that need to be hashed

Definition at line 136 of file functions.cpp.

References hasher.

Referenced by BOOST\_AUTO\_TEST\_CASE(), and main().

```

138 {
139     auto start = std::chrono::steady_clock::now();
140     size_t size = hashes.size();
141     if (size == words.size() - 2)
142     {
143         for (size_t i = 0; i < size; ++i)
144         {
145             hashes[i] = hasher(words[i] + "-" +
146                 words[i + 1] + "-" +
147                 words[i + 2]);
148         }
149     }
150     else if (size == words.size() - 1)
151     {
152         for (size_t i = 0; i < size; ++i)
153         {
154             hashes[i] = hasher(words[i] + "-" +
155                 words[i + 1]);
156         }
157     }
```

```

157     }
158     else
159     {
160         for (size_t i = 0; i < size; ++i)
161         {
162             hashes[i] = hasher(words[i]);
163         }
164     }
165
166     auto end = std::chrono::steady_clock::now();
167     BOOST_LOG_TRIVIAL(debug) << "init_hashes() "
168         << std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count() << " ms";
169 }

```

Here is the caller graph for this function:

### 2.2.1.8 init\_logging()

```
void init_logging ( )
```

This function helps in logging level.

Definition at line 22 of file functions.cpp.

Referenced by main().

```

23 {
24     logging::register_simple_formatter_factory<logging::trivial::severity_level, char>("Severity");
25
26     logging::add_file_log(
27         keywords::file_name = "logfile.log",
28         keywords::format = "[%TimeStamp%] [%ThreadID%] [%Severity%] [%ProcessID%] [%LineID%] %Message%"
29     );
30
31     logging::core::get()->set_filter
32     (
33         logging::trivial::severity >= logging::trivial::trace
34     );
35
36     logging::add_common_attributes();
37 }

```

Here is the caller graph for this function:

### 2.2.1.9 init\_random\_matrices() [1/2]

```
void init_random_matrices (
    std::vector< std::bitset< 100 >> & matrices )
```

initializing random matrices

#### Parameters

<i>matrices</i>	- vector of bitsets, contain all initialized matrices of length 100
-----------------	---

Definition at line 39 of file functions.cpp.

References dict\_size, and matrices().

Referenced by BOOST\_AUTO\_TEST\_CASE(), and main().

```

40 {
41     auto start = std::chrono::steady_clock::now();
42     size_t dict_size = matrices.size();
43     std::default_random_engine dre(std::random_device{}());
44     std::uniform_int_distribution<long long> dist(0, (1ll << 50) - 1);
45     for (size_t i = 0; i < dict_size; ++i)
46     {
47         matrices[i] = dist(dre);
48         matrices[i] <= 50;
49         matrices[i] |= dist(dre);
50     }
51     auto end = std::chrono::steady_clock::now();
52     BOOST_LOG_TRIVIAL(debug) << "init_random_matrices() "
53         << std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count() << " ms";
54 }

```

Here is the call graph for this function: Here is the caller graph for this function:

### 2.2.1.10 init\_random\_matrices() [2/2]

```

void init_random_matrices (
    std::vector< std::vector< size_t >> & matrices )

```

Definition at line 470 of file functions.cpp.

References `dict_size`, `matrices()`, and `matrix_size`.

```

471 {
472     size_t dict_size = matrices.size();
473     size_t matrix_size = matrices[0].size();
474
475     for (size_t i = 0; i < dict_size; ++i)
476     {
477         for (size_t j = 0; j < matrix_size; ++j)
478         {
479             matrices[i][j] = rand() % 2;
480         }
481     }
482 }

```

Here is the call graph for this function:

### 2.2.1.11 print\_bitset()

```

void print_bitset (
    const std::bitset< 100 > & bitset )

```

Definition at line 405 of file functions.cpp.

```

406 {
407     for (size_t j = 0; j < 10; ++j)
408     {
409         for (size_t i = j * 10; i < j * 10 + 10; ++i)
410         {
411             std::cout << bitset[i] << " ";
412         }
413         std::cout << std::endl;
414     }
415 }

```



### 2.2.1.12 print\_bitset\_vector()

```
void print_bitset_vector (
    const std::bitset< 100 > & bitset )
```

Definition at line 417 of file functions.cpp.

```
418 {
419     for (size_t j = 0; j < bitset.size(); ++j)
420     {
421         std::cout << bitset[j];
422     }
423     std::cout << std::endl;
424 }
```

### 2.2.1.13 print\_duplicates()

```
void print_duplicates (
    const std::vector< std::vector< size_t >> & duplicates,
    const std::vector< std::string > & dictionary )
```

Definition at line 354 of file functions.cpp.

References [duplicates](#).

```
356 {
357     for (int i = 0; i < duplicates.size(); ++i)
358     {
359         for (int j = 0; j < duplicates[0].size(); ++j)
360         {
361             if (j == 1)
362             {
363                 std::cout << "dictionary[" << duplicates[i][j] << "] = " << dictionary[duplicates[i][j]] <<
364                 " ";
365             }
366             else
367             {
368                 std::cout << duplicates[i][j] << " ";
369             }
370         }
371         std::cout << std::endl;
372     }
373 }
```

### 2.2.1.14 print\_hash\_table()

```
void print_hash_table (
    const std::vector< std::pair< size_t, size_t >> & hash_table,
    size_t begin = 0,
    size_t end = 466548 )
```

Definition at line 345 of file functions.cpp.

References [hash\\_table\(\)](#).

```
346 {
347     for (size_t i = begin; i < end; ++i)
348     {
349         //std::cout << "hash = " << hash_table[i].first << " index = " << hash_table[i].second <<
350         std::endl;
351         std::cout << hash_table[i].first << " " << hash_table[i].second << std::endl;
352     }
```

Here is the call graph for this function:

### 2.2.1.15 print\_matrix()

```
void print_matrix (
    const std::vector< size_t > & matrix )
```

Definition at line 375 of file functions.cpp.

```
376 {
377     for (size_t j = 0; j < 10; ++j)
378     {
379         for (size_t i = j * 10; i < j * 10 + 10; ++i)
380         {
381             std::cout << matrix[i] << " ";
382         }
383         std::cout << std::endl;
384     }
385 }
```

### 2.2.1.16 print\_vector()

```
void print_vector (
    const std::vector< size_t > & vector )
```

Definition at line 396 of file functions.cpp.

```
397 {
398     for (size_t i = 0; i < vector.size(); ++i)
399     {
400         std::cout << vector[i];
401     }
402     std::cout << std::endl;
403 }
```

### 2.2.1.17 print\_words()

```
void print_words (
    const std::vector< std::string > & dictionary )
```

Definition at line 387 of file functions.cpp.

```
388 {
389     for (size_t i = 0; i < dictionary.size(); ++i)
390     {
391         std::cout << dictionary[i] << std::endl;
392     }
393     std::cout << std::endl;
394 }
```

### 2.2.1.18 read\_dict\_from\_file()

```
void read_dict_from_file (
    const std::string & dictionary_filename,
    std::vector< std::string > & dictionary )
```

reading dictionary from file

## Parameters

<i>dictionary_filename</i>	- the filename that need to be read
<i>dictionary</i>	- contain all words in the file

Definition at line 56 of file functions.cpp.

Referenced by BOOST\_AUTO\_TEST\_CASE(), and main().

```

57 {
58     auto start = std::chrono::steady_clock::now();
59     std::ifstream file(dictionary_filename);
60     size_t counter = 0;
61     if (file.is_open())
62     {
63         std::string line;
64         size_t i = 0;
65         while (std::getline(file, line))
66         {
67             dictionary[i++] = line;
68             counter++;
69         }
70         dictionary.resize(counter);
71         file.close();
72     }
73     else
74     {
75         BOOST_LOG_TRIVIAL(error) << "Couldn't open " << dictionary_filename << " for
reading";
76         std::cerr << "Couldn't open " << dictionary_filename << " for reading\n";
77     }
78     auto end = std::chrono::steady_clock::now();
79     BOOST_LOG_TRIVIAL(debug) << "read_dict_from_file() "
80         << std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count() << " ms";
81 }
```

Here is the caller graph for this function:

## 2.2.1.19 read\_text\_from\_file()

```

void read_text_from_file (
    const std::string & text_filename,
    std::vector< std::string > & text )
```

reading input text, stop words from file

## Parameters

<i>text_filename</i>	- the filename that need to be read
<i>text</i>	- contain all words in the file

Definition at line 83 of file functions.cpp.

Referenced by BOOST\_AUTO\_TEST\_CASE(), and main().

```

84 {
85     auto start = std::chrono::steady_clock::now();
86     std::ifstream file(text_filename);
87     if (file.is_open())
88     {
89         std::string line;
90         size_t i = 0, counter = 0;
```

```

91     while (std::getline(file, line))
92     {
93         //std::cout << line << " " << line.size() << std::endl;
94         size_t size = line.size();
95         if(line[size - 1] < 'A' || line[size - 1] > 'Z' &&
96            line[size - 1] < 'a' || line[size - 1] > 'z')
97         {
98             line.pop_back();
99         }
100
101         char* str = &line[0];
102         char* pch;
103         pch = strtok(str, " ,.-?");
104
105         //std::cout << line << " " << line.size() << std::endl;
106
107         while (pch != NULL)
108         {
109             for (size_t j = 0; j < strlen(pch); j++)
110             {
111                 if (pch[j] >= 65 && pch[j] <= 92)
112                 {
113                     pch[j] = pch[j] + 32;
114                 }
115             }
116             //std::cout << pch << " " << strlen(pch) << std::endl;
117             text[i++] = pch;
118             pch = strtok(NULL, " ,.-?");
119             counter++;
120         }
121     }
122     text.resize(counter);
123     file.close();
124 }
125 else
126 {
127     BOOST_LOG_TRIVIAL(error) << "Couldn't open " << text_filename << " for reading";
128     std::cerr << "Couldn't open " << text_filename << " for reading\n";
129 }
130
131 auto end = std::chrono::steady_clock::now();
132 BOOST_LOG_TRIVIAL(debug) << "read_from_text_file() "
133    << std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count() << " ms";
134 }

```

Here is the caller graph for this function:

### 2.2.1.20 search\_and\_calculate\_matrices()

```

void search_and_calculate_matrices (
    std::bitset< 100 > & output,
    std::map< std::string, size_t > & words_count,
    const std::vector< size_t > & text_single_term_hashes,
    const std::vector< size_t > & text_double_term_hashes,
    const std::vector< size_t > & text_triple_term_hashes,
    const std::vector< size_t > & stop_words_hashes,
    const std::vector< std::pair< size_t, size_t >> & hash_table,
    const std::vector< std::bitset< 100 >> & matrices,
    const std::vector< std::string > & dictionary )

```

correcting input text hashes, maybe there are duplicates

#### Parameters

<i>output</i>	- the result of the program, score of the sentence
<i>text_single_term_hashes</i>	- hash of text, contain hashed strings of each word in text
<i>text_double_term_hashes</i>	- hash of text, contain hashed strings of each double combination of words in text
<i>stop_words_hashes</i>	- contain all hashes of stop words
<i>hash_table</i>	- hash table contain hashes and indexes to the words in dictionary
<i>matrices</i>	- vector of bitsets, contain all initialized matrices of length 100
<i>dictionary</i>	- contain all words in dictionary.txt

Definition at line 255 of file functions.cpp.

References `dict_size`, `hash_table()`, `matrices()`, `matrix_size`, `text_double_term_hashes()`, `text_single_term_hashes()`, `text_size`, and `text_triple_term_hashes()`.

Referenced by `BOOST_AUTO_TEST_CASE()`, and `main()`.

```

264 {
265     auto start = std::chrono::steady_clock::now();
266     size_t text_size = text_single_term_hashes.size();
267     size_t dict_size = dictionary.size();
268     size_t matrix_size = output.size();
269     std::vector<size_t> indexes(text_size);
270     std::vector<const std::vector<size_t*>> terms(3);
271     terms[0] = &text_single_term_hashes;
272     terms[1] = &text_double_term_hashes;
273     terms[2] = &text_triple_term_hashes;
274     for(size_t k = 3; k > 0; --k)
275     {
276         for (size_t i = 0; i < text_size - k + 1; ++i)
277         {
278             bool b = true;
279             for(size_t l = 0; l < k; ++l)
280             {
281                 if (indexes[i + l])
282                 {
283                     b = false;
284                 }
285             }
286             if(k == 1)
287             {
288                 for (size_t l = 0; l < stop_words_hashes.size(); ++l)
289                 {
290                     if (stop_words_hashes[l] == (*terms[0])[i])
291                     {
292                         indexes[i] = 1;
293                         b = false;
294                     }
295                 }
296             }
297             if (b)
298             {
299                 size_t index = ((*terms[k - 1])[i]) % dict_size;
300                 for (size_t j = 0; j < dict_size; ++j)
301                 {
302                     if (hash_table[index].first == ((*terms[k - 1])[i]))
303                     {
304                         for(size_t l = 0; l < k; ++l)
305                         {
306                             indexes[i + l] = 1;
307                         }
308                         output |= matrices[hash_table[index].second];
309                         size_t& value = words_count[dictionary[hash_table[index].second]];
310                         value? value++ : value = 1;
311                         break;
312                     }
313                     else
314                     {
315                         index = (index + 1) % dict_size;
316                     }
317                 }
318             }
319         }
320     }
321     auto end = std::chrono::steady_clock::now();
322     BOOST_LOG_TRIVIAL(debug) << "search_and_calculate_matrices() "
323     << std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count() << " ms";
324 }
325 }
```

Here is the call graph for this function: Here is the caller graph for this function:

### 2.2.1.21 write\_in\_file()

```
void write_in_file ( )
```

## 2.3 src/main.cpp File Reference

```
#include "functions.hpp"
#include <iostream>
#include <time.h>
#include <string.h>
#include <chrono>
#include <map>
#include <set>
#include <algorithm>
#include <functional>
#include <boost/log/trivial.hpp>
Include dependency graph for main.cpp:
```

### Macros

- `#define max_dict_size 500000`
- `#define max_term_size 20000`
- `#define max_stop_words_size 200`
- `#define matrix_size 100`

### Functions

- `int main ()`

### 2.3.1 Macro Definition Documentation

#### 2.3.1.1 matrix\_size

```
#define matrix_size 100
```

Definition at line 15 of file main.cpp.

Referenced by `init_random_matrices()`, `main()`, and `search_and_calculate_matrices()`.

#### 2.3.1.2 max\_dict\_size

```
#define max_dict_size 500000
```

Definition at line 12 of file main.cpp.

Referenced by `main()`.

### 2.3.1.3 max\_stop\_words\_size

```
#define max_stop_words_size 200
```

Definition at line 14 of file main.cpp.

Referenced by main().

### 2.3.1.4 max\_term\_size

```
#define max_term_size 20000
```

Definition at line 13 of file main.cpp.

Referenced by main().

## 2.3.2 Function Documentation

### 2.3.2.1 main()

```
int main ( )
```

Definition at line 17 of file main.cpp.

References `correct_dict_hashes_and_extract_duplicates()`, `correct_text_hashes()`, `dict_hashes()`, `dict_size`, `dictionary()`, `dictionary_filename`, `duplicates`, `duplicates_size`, `extract_words_count()`, `hash_function()`, `hash_table()`, `hasher`, `init_hashes()`, `init_logging()`, `init_random_matrices()`, `matrices()`, `matrix_size`, `max_dict_size`, `max_stop_words_size`, `max_term_size`, `output()`, `read_dict_from_file()`, `read_text_from_file()`, `search_and_calculate_matrices()`, `stop_words()`, `stop_words_filename`, `stop_words_hashes()`, `stop_words_size`, `text()`, `text_double_term_hashes()`, `text_filename`, `text_single_term_hashes()`, `text_size`, `text_triple_term_hashes()`, and `words_count`.

```
18 {
19     init_logging();
20
21     auto start = std::chrono::steady_clock::now();
22     const std::string dictionary_filename = "dictionary.txt";
23     std::vector<std::string> dictionary(max_dict_size);
24     read_dict_from_file(dictionary_filename, dictionary);
25     size_t dict_size = dictionary.size();
26
27     const std::string text_filename = "text2.txt";
28     std::vector<std::string> text(max_term_size);
29     read_text_from_file(text_filename, text);
30     size_t text_size = text.size();
31
32     const std::string stop_words_filename = "stop_words.txt";
33     std::vector<std::string> stop_words(max_stop_words_size);
34     read_text_from_file(stop_words_filename, stop_words);
35     size_t stop_words_size = stop_words.size();
36
37     std::hash<std::string> hasher;
38     std::vector<std::vector<size_t>> duplicates;
39
40     std::vector<size_t> dict_hashes(dict_size);
41     init_hashes(dict_hashes, hasher, dictionary);
42     correct_dict_hashes_and_extract_duplicates(duplicates,
dict_hashes);
```

```

43     size_t duplicates_size = duplicates.size();
44
45     std::vector<size_t> text_single_term_hashes(text_size);
46     init_hashes(text_single_term_hashes, hasher,
47 text);
48     if(duplicates_size)
49     {
50         correct_text_hashes(text_single_term_hashes,
51 text, duplicates, dictionary);
52     }
53     std::vector<size_t> text_double_term_hashes(text_size - 1);
54     init_hashes(text_double_term_hashes, hasher,
55 text);
56     if(duplicates_size)
57     {
58         correct_text_hashes(text_double_term_hashes,
59 text, duplicates, dictionary);
60     }
61     std::vector<size_t> text_triple_term_hashes(text_size - 2);
62     init_hashes(text_triple_term_hashes, hasher,
63 text);
64     if(duplicates_size)
65     {
66         correct_text_hashes(text_triple_term_hashes,
67 text, duplicates, dictionary);
68     }
69     std::vector<size_t> stop_words_hashes(stop_words_size);
70     init_hashes(stop_words_hashes, hasher,
71 stop_words);
72     if(duplicates_size)
73     {
74         correct_text_hashes(stop_words_hashes,
75 stop_words, duplicates, dictionary);
76     }
77     std::vector<std::bitset<100>> matrices(dict_size);
78     init_random_matrices(matrices);
79
80     std::vector<std::pair<size_t, size_t>> hash_table(dict_size);
81     hash_function(hash_table, dict_hashes);
82
83     std::map<std::string, size_t> words_count;
84     std::bitset<100> output(matrix_size);
85     search_and_calculate_matrices(output, words_count,
86 text_single_term_hashes, text_double_term_hashes,
87 text_triple_term_hashes, stop_words_hashes,
88 hash_table, matrices, dictionary);
89
90     extract_words_count(words_count);
91     BOOST_LOG_TRIVIAL(info) << "Dictionary size = " << dict_size;
92     BOOST_LOG_TRIVIAL(debug) << "Duplicates count = " << duplicates.size();
93     BOOST_LOG_TRIVIAL(info) << "Stop words count = " << stop_words.size();
94     BOOST_LOG_TRIVIAL(info) << "Input words count = " << text_single_term_hashes.
95 size();
96     BOOST_LOG_TRIVIAL(info) << "The result is = " << output;
97
98     auto end = std::chrono::steady_clock::now();
99     BOOST_LOG_TRIVIAL(debug) << "Elapsed time in milliseconds : "
100 << std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count() << " ms";
101     return 0;
102 }

```

Here is the call graph for this function:

## 2.4 src/unit\_tests.cpp File Reference

```

#include "functions.hpp"
#include <unordered_set>
#include <bitset>
#include <boost/test/unit_test.hpp>
#include <boost/log/trivial.hpp>
#include <boost/log/core.hpp>
#include <boost/log/expressions.hpp>

```



```
#include <boost/log/utility/setup/file.hpp>
#include <boost/log/utility/setup/common_attributes.hpp>
Include dependency graph for unit_tests.cpp:
```

## Macros

- #define [BOOST\\_TEST\\_MODULE](#) functions
- #define [max\\_dict\\_size](#) 500000
- #define [max\\_term\\_size](#) 20000
- #define [max\\_stop\\_words\\_size](#) 200
- #define [matrix\\_size](#) 100

## Functions

- `std::vector< std::string >` [dictionary](#) (500000)
- `std::vector< size_t >` [dict\\_hashes](#) (500000)
- `std::vector< std::string >` [text](#) (20000)
- `std::vector< size_t >` [text\\_single\\_term\\_hashes](#) (20000)
- `std::vector< size_t >` [text\\_double\\_term\\_hashes](#) (20000)
- `std::vector< size_t >` [text\\_triple\\_term\\_hashes](#) (20000)
- `std::vector< std::string >` [stop\\_words](#) (200)
- `std::vector< size_t >` [stop\\_words\\_hashes](#) (200)
- `std::vector< std::pair< size_t, size_t > >` [hash\\_table](#) (500000)
- `std::vector< std::bitset< 100 > >` [matrices](#) (500000)
- `std::bitset< 100 >` [output](#) (100)
- [BOOST\\_AUTO\\_TEST\\_CASE](#) (read\_dict\_from\_file\_test)
- [BOOST\\_AUTO\\_TEST\\_CASE](#) (read\_text\_from\_file\_test)
- [BOOST\\_AUTO\\_TEST\\_CASE](#) (init\_random\_matrices\_test)
- [BOOST\\_AUTO\\_TEST\\_CASE](#) (init\_hashes\_test)
- [BOOST\\_AUTO\\_TEST\\_CASE](#) (correct\_dict\_hashes\_and\_extract\_duplicates\_test)
- [BOOST\\_AUTO\\_TEST\\_CASE](#) (hash\_function\_test)
- [BOOST\\_AUTO\\_TEST\\_CASE](#) (search\_and\_calculate\_matrices\_test)

## Variables

- `std::hash< std::string >` [hasher](#)
- `const std::string` [dictionary\\_filename](#) = "dictionary.txt"
- `size_t` [dict\\_size](#)
- `const std::string` [text\\_filename](#) = "text1.txt"
- `size_t` [text\\_size](#)
- `const std::string` [stop\\_words\\_filename](#) = "stop\_words.txt"
- `size_t` [stop\\_words\\_size](#)
- `std::vector< std::vector< size_t > >` [duplicates](#)
- `size_t` [duplicates\\_size](#)
- `std::map< std::string, size_t >` [words\\_count](#)

### 2.4.1 Macro Definition Documentation

#### 2.4.1.1 BOOST\_TEST\_MODULE

```
#define BOOST_TEST_MODULE functions
```

Definition at line 1 of file unit\_tests.cpp.

#### 2.4.1.2 matrix\_size

```
#define matrix_size 100
```

Definition at line 17 of file unit\_tests.cpp.

#### 2.4.1.3 max\_dict\_size

```
#define max_dict_size 500000
```

Definition at line 14 of file unit\_tests.cpp.

Referenced by BOOST\_AUTO\_TEST\_CASE().

#### 2.4.1.4 max\_stop\_words\_size

```
#define max_stop_words_size 200
```

Definition at line 16 of file unit\_tests.cpp.

#### 2.4.1.5 max\_term\_size

```
#define max_term_size 20000
```

Definition at line 15 of file unit\_tests.cpp.

Referenced by BOOST\_AUTO\_TEST\_CASE().

### 2.4.2 Function Documentation

## 2.4.2.1 BOOST\_AUTO\_TEST\_CASE() [1/7]

```
BOOST_AUTO_TEST_CASE (
    read_dict_from_file_test )
```

Definition at line 47 of file unit\_tests.cpp.

References `dict_size`, `dictionary()`, `dictionary_filename`, `max_dict_size`, and `read_dict_from_file()`.

```
48 {
49     read_dict_from_file(dictionary_filename,
50 dictionary);
51     dict_size = dictionary.size();
52     BOOST_REQUIRE_LE( dict_size, max_dict_size);
53     for(size_t i = 0; i < dict_size; ++i)
54     {
55         BOOST_CHECK_NE(dictionary[i].size(), 0);
56         for(size_t j = 0; j < dictionary[i].size(); ++j)
57         {
58             // Decimal 32 = [Space] character
59             // Decimal 127 = [DEL] character
60             // (32, 127) in this area characters are visible
61             BOOST_WARN(dictionary[i][j] > 32 &&
62 dictionary[i][j] < 127);
63         }
64 }
```

Here is the call graph for this function:

## 2.4.2.2 BOOST\_AUTO\_TEST\_CASE() [2/7]

```
BOOST_AUTO_TEST_CASE (
    read_text_from_file_test )
```

Definition at line 66 of file unit\_tests.cpp.

References `max_term_size`, `read_text_from_file()`, `stop_words()`, `stop_words_filename`, `stop_words_size`, `text()`, `text_filename`, and `text_size`.

```
67 {
68     read_text_from_file(text_filename, text);
69     text_size = text.size();
70     BOOST_REQUIRE_LE( text_size, max_term_size);
71     for(size_t i = 0; i < text_size; ++i)
72     {
73         BOOST_CHECK_NE(text[i].size(), 0);
74         for(size_t j = 0; j < text[i].size(); ++j)
75         {
76             // Decimal 32 = [Space] character
77             // Decimal 127 = [DEL] character
78             // (32, 127) in this area characters are visible
79             BOOST_WARN(text[i][j] > 32 &&
80 text[i][j] < 127);
81         }
82     }
83
84     read_text_from_file(stop_words_filename,
85 stop_words);
86     stop_words_size = stop_words.size();
87     BOOST_REQUIRE_LE( stop_words_size, max_term_size);
88     for(size_t i = 0; i < stop_words_size; ++i)
89     {
90         BOOST_CHECK_NE(stop_words[i].size(), 0);
91         for(size_t j = 0; j < stop_words[i].size(); ++j)
92         {
93             // Decimal 32 = [Space] character
94             // Decimal 127 = [DEL] character
95             // (32, 127) in this area characters are visible
96             BOOST_WARN(stop_words[i][j] > 32 &&
97 stop_words[i][j] < 127);
98         }
99 }
```

Here is the call graph for this function:

**2.4.2.3 BOOST\_AUTO\_TEST\_CASE()** [3/7]

```
BOOST_AUTO_TEST_CASE (
    init_random_matrices_test )
```

Definition at line 101 of file unit\_tests.cpp.

References dict\_size, init\_random\_matrices(), and matrices().

```
102 {
103     matrices.resize(dict_size);
104     init_random_matrices(matrices);
105     size_t size = matrices.size();
106     std::unordered_set<std::bitset<matrix_size>> un_set(matrices.begin(),
107     matrices.end());
107     BOOST_WARN_EQUAL(un_set.size(), size);
108 }
```

Here is the call graph for this function:

**2.4.2.4 BOOST\_AUTO\_TEST\_CASE()** [4/7]

```
BOOST_AUTO_TEST_CASE (
    init_hashes_test )
```

Definition at line 110 of file unit\_tests.cpp.

References dict\_hashes(), dict\_size, dictionary(), hasher, init\_hashes(), stop\_words(), stop\_words\_hashes(), stop\_words\_size, text(), text\_double\_term\_hashes(), text\_single\_term\_hashes(), text\_size, and text\_triple\_term\_hashes().

```
111 {
112     dict_hashes.resize(dict_size);
113     init_hashes(dict_hashes, hasher, dictionary);
114     BOOST_CHECK_EQUAL(dict_hashes.size(), dict_size);
115     size_t size = dict_hashes.size();
116     std::unordered_set<std::bitset<matrix_size>> un_set(dict_hashes.begin(),
117     dict_hashes.end());
117     BOOST_WARN_EQUAL( un_set.size(), size );
118     un_set.clear();
119
120     text_single_term_hashes.resize(text_size);
121     init_hashes(text_single_term_hashes,
122     hasher, text);
122     BOOST_WARN_EQUAL(text_single_term_hashes.size(),
123     text_size);
123     size = text_single_term_hashes.size();
124     un_set = std::unordered_set<std::bitset<matrix_size>>(
125     text_single_term_hashes.begin(),text_single_term_hashes.end()
126     );
125     BOOST_WARN_LE( un_set.size(), size );
126     un_set.clear();
127
128
129     text_double_term_hashes.resize(text_size - 1);
130     init_hashes(text_double_term_hashes,
131     hasher, text);
131     BOOST_CHECK_EQUAL(text_double_term_hashes.size(),
132     text_size - 1);
132     size = text_double_term_hashes.size();
133     un_set = std::unordered_set<std::bitset<matrix_size>>(
134     text_double_term_hashes.begin(),text_double_term_hashes.end()
135     );
134     BOOST_WARN_LE( un_set.size(), size );
135     un_set.clear();
136
137     text_triple_term_hashes.resize(text_size - 2);
138     init_hashes(text_triple_term_hashes,
139     hasher, text);
139     BOOST_CHECK_EQUAL(text_triple_term_hashes.size(),
```

```

    text_size - 2);
140     size = text_triple_term_hashes.size();
141     un_set = std::unordered_set<std::bitset<matrix_size>>(
    text_triple_term_hashes.begin(), text_triple_term_hashes.end()
    );
142     BOOST_WARN_LE( un_set.size(), size );
143     un_set.clear();
144
145     stop_words_hashes.resize(stop_words_size);
146     init_hashes(stop_words_hashes, hasher,
    stop_words);
147     BOOST_CHECK_EQUAL(stop_words_hashes.size(), stop_words_size);
148     size = stop_words_hashes.size();
149     un_set = std::unordered_set<std::bitset<matrix_size>>(stop_words_hashes.begin(),
    stop_words_hashes.end());
150     BOOST_WARN_EQUAL( un_set.size(), size );
151 }

```

Here is the call graph for this function:

#### 2.4.2.5 BOOST\_AUTO\_TEST\_CASE() [5/7]

```

BOOST_AUTO_TEST_CASE (
    correct_dict_hashes_and_extract_duplicates_test )

```

Definition at line 153 of file unit\_tests.cpp.

References `correct_dict_hashes_and_extract_duplicates()`, `dict_hashes()`, `dict_size`, and `duplicates_size`.

```

154 {
155     correct_dict_hashes_and_extract_duplicates(
    duplicates, dict_hashes);
156     size_t duplicates_size = duplicates.size();
157     BOOST_CHECK_EQUAL(duplicates_size, 0);
158     BOOST_CHECK_EQUAL(dict_hashes.size(), dict_size);
159     size_t size = dict_hashes.size();
160     std::unordered_set<std::bitset<matrix_size>> un_set(dict_hashes.begin(),
    dict_hashes.end());
161     BOOST_WARN_EQUAL( un_set.size(), size );
162 }

```

Here is the call graph for this function:

#### 2.4.2.6 BOOST\_AUTO\_TEST\_CASE() [6/7]

```

BOOST_AUTO_TEST_CASE (
    hash_function_test )

```

Definition at line 164 of file unit\_tests.cpp.

References `dict_hashes()`, `dict_size`, `hash_function()`, and `hash_table()`.

```

165 {
166     hash_table.resize(dict_size);
167     hash_function(hash_table, dict_hashes);
168     BOOST_CHECK_EQUAL(hash_table.size(), dict_hashes.size());
169     for(size_t i = 0; i < dict_size; ++i)
170     {
171         BOOST_CHECK_NE(hash_table[i].first, 0);
172     }
173 }

```

Here is the call graph for this function:

### 2.4.2.7 BOOST\_AUTO\_TEST\_CASE() [7/7]

```
BOOST_AUTO_TEST_CASE (
    search_and_calculate_matrices_test )
```

Definition at line 175 of file unit\_tests.cpp.

References dictionary(), hash\_table(), matrices(), output(), search\_and\_calculate\_matrices(), stop\_words\_  
hashes(), text\_double\_term\_hashes(), text\_single\_term\_hashes(), and text\_triple\_term\_hashes().

```
176 {
177     std::bitset<matrix_size> temp = output;
178     search_and_calculate_matrices(output,
179     words_count, text_single_term_hashes,
180     text_double_term_hashes,
181     text_triple_term_hashes, stop_words_hashes,
182     hash_table, matrices, dictionary);
183     BOOST_CHECK_NE(temp, output);
184 }
```

Here is the call graph for this function:

### 2.4.2.8 dict\_hashes()

```
std::vector<size_t> dict_hashes (
    500000 )
```

Referenced by BOOST\_AUTO\_TEST\_CASE(), and main().

Here is the caller graph for this function:

### 2.4.2.9 dictionary()

```
std::vector<std::string> dictionary (
    500000 )
```

Referenced by BOOST\_AUTO\_TEST\_CASE(), and main().

Here is the caller graph for this function:

### 2.4.2.10 hash\_table()

```
std::vector<std::pair<size_t,size_t> > hash_table (
    500000 )
```

Referenced by BOOST\_AUTO\_TEST\_CASE(), hash\_function(), main(), print\_hash\_table(), and search\_and\_  
calculate\_matrices().

Here is the caller graph for this function:

#### 2.4.2.11 matrices()

```
std::vector<std::bitset< 100 > > matrices (
    500000 )
```

Referenced by BOOST\_AUTO\_TEST\_CASE(), init\_random\_matrices(), main(), and search\_and\_calculate\_matrices().

Here is the caller graph for this function:

#### 2.4.2.12 output()

```
std::bitset< 100 > output (
    100 )
```

Referenced by BOOST\_AUTO\_TEST\_CASE(), and main().

Here is the caller graph for this function:

#### 2.4.2.13 stop\_words()

```
std::vector<std::string> stop_words (
    200 )
```

Referenced by BOOST\_AUTO\_TEST\_CASE(), and main().

Here is the caller graph for this function:

#### 2.4.2.14 stop\_words\_hashes()

```
std::vector<size_t> stop_words_hashes (
    200 )
```

Referenced by BOOST\_AUTO\_TEST\_CASE(), and main().

Here is the caller graph for this function:

#### 2.4.2.15 text()

```
std::vector<std::string> text (
    20000 )
```

Referenced by BOOST\_AUTO\_TEST\_CASE(), and main().

Here is the caller graph for this function:

#### 2.4.2.16 text\_double\_term\_hashes()

```
std::vector<size_t> text_double_term_hashes (
    20000 )
```

Referenced by BOOST\_AUTO\_TEST\_CASE(), main(), and search\_and\_calculate\_matrices().

Here is the caller graph for this function:

#### 2.4.2.17 text\_single\_term\_hashes()

```
std::vector<size_t> text_single_term_hashes (
    20000 )
```

Referenced by BOOST\_AUTO\_TEST\_CASE(), main(), and search\_and\_calculate\_matrices().

Here is the caller graph for this function:

#### 2.4.2.18 text\_triple\_term\_hashes()

```
std::vector<size_t> text_triple_term_hashes (
    20000 )
```

Referenced by BOOST\_AUTO\_TEST\_CASE(), main(), and search\_and\_calculate\_matrices().

Here is the caller graph for this function:

### 2.4.3 Variable Documentation

#### 2.4.3.1 dict\_size

```
size_t dict_size
```

Definition at line 24 of file unit\_tests.cpp.

Referenced by BOOST\_AUTO\_TEST\_CASE(), init\_random\_matrices(), main(), and search\_and\_calculate\_matrices().

#### 2.4.3.2 dictionary\_filename

```
const std::string dictionary_filename = "dictionary.txt"
```

Definition at line 21 of file unit\_tests.cpp.

Referenced by BOOST\_AUTO\_TEST\_CASE(), and main().



### 2.4.3.3 duplicates

```
std::vector<std::vector<size_t> > duplicates
```

Definition at line 38 of file unit\_tests.cpp.

Referenced by `correct_dict_hashes_and_extract_duplicates()`, `correct_text_hashes()`, `main()`, and `print_duplicates()`.

### 2.4.3.4 duplicates\_size

```
size_t duplicates_size
```

Definition at line 39 of file unit\_tests.cpp.

Referenced by `BOOST_AUTO_TEST_CASE()`, and `main()`.

### 2.4.3.5 hasher

```
std::hash<std::string> hasher
```

Definition at line 19 of file unit\_tests.cpp.

Referenced by `BOOST_AUTO_TEST_CASE()`, `init_hashes()`, and `main()`.

### 2.4.3.6 stop\_words\_filename

```
const std::string stop_words_filename = "stop_words.txt"
```

Definition at line 33 of file unit\_tests.cpp.

Referenced by `BOOST_AUTO_TEST_CASE()`, and `main()`.

### 2.4.3.7 stop\_words\_size

```
size_t stop_words_size
```

Definition at line 36 of file unit\_tests.cpp.

Referenced by `BOOST_AUTO_TEST_CASE()`, and `main()`.

#### 2.4.3.8 text\_filename

```
const std::string text_filename = "text1.txt"
```

Definition at line 26 of file unit\_tests.cpp.

Referenced by BOOST\_AUTO\_TEST\_CASE(), and main().

#### 2.4.3.9 text\_size

```
size_t text_size
```

Definition at line 31 of file unit\_tests.cpp.

Referenced by BOOST\_AUTO\_TEST\_CASE(), correct\_text\_hashes(), main(), and search\_and\_calculate\_matrices().

#### 2.4.3.10 words\_count

```
std::map<std::string, size_t> words_count
```

Definition at line 43 of file unit\_tests.cpp.

Referenced by hash\_function(), and main().

# Index

## BOOST\_AUTO\_TEST\_CASE

unit\_tests.cpp, 20–23

## BOOST\_TEST\_MODULE

unit\_tests.cpp, 19

## contains\_duplicates

functions.hpp, 4

## contains\_duplicates\_h

functions.hpp, 5

## correct\_dict\_hashes\_and\_extract\_duplicates

functions.hpp, 5

## correct\_text\_hashes

functions.hpp, 6

## dict\_hashes

unit\_tests.cpp, 24

## dict\_size

unit\_tests.cpp, 26

## dictionary

unit\_tests.cpp, 24

## dictionary\_filename

unit\_tests.cpp, 26

## duplicates

unit\_tests.cpp, 26

## duplicates\_size

unit\_tests.cpp, 27

## extract\_words\_count

functions.hpp, 7

## functions.hpp

contains\_duplicates, 4

contains\_duplicates\_h, 5

correct\_dict\_hashes\_and\_extract\_duplicates, 5

correct\_text\_hashes, 6

extract\_words\_count, 7

hash\_function, 7

init\_hashes, 8

init\_logging, 9

init\_random\_matrices, 9, 10

print\_bitset, 10

print\_bitset\_vector, 10

print\_duplicates, 11

print\_hash\_table, 11

print\_matrix, 11

print\_vector, 12

print\_words, 12

read\_dict\_from\_file, 12

read\_text\_from\_file, 13

search\_and\_calculate\_matrices, 14

write\_in\_file, 15

## hash\_function

functions.hpp, 7

## hash\_table

unit\_tests.cpp, 24

## hasher

unit\_tests.cpp, 27

## init\_hashes

functions.hpp, 8

## init\_logging

functions.hpp, 9

## init\_random\_matrices

functions.hpp, 9, 10

## main

main.cpp, 17

## main.cpp

main, 17

matrix\_size, 16

max\_dict\_size, 16

max\_stop\_words\_size, 16

max\_term\_size, 17

## matrices

unit\_tests.cpp, 24

## matrix\_size

main.cpp, 16

unit\_tests.cpp, 20

## max\_dict\_size

main.cpp, 16

unit\_tests.cpp, 20

## max\_stop\_words\_size

main.cpp, 16

unit\_tests.cpp, 20

## max\_term\_size

main.cpp, 17

unit\_tests.cpp, 20

## output

unit\_tests.cpp, 25

## print\_bitset

functions.hpp, 10

## print\_bitset\_vector

functions.hpp, 10

## print\_duplicates

functions.hpp, 11

## print\_hash\_table

functions.hpp, 11

## print\_matrix

- functions.hpp, 11
- print\_vector
  - functions.hpp, 12
- print\_words
  - functions.hpp, 12
- read\_dict\_from\_file
  - functions.hpp, 12
- read\_text\_from\_file
  - functions.hpp, 13
- search\_and\_calculate\_matrices
  - functions.hpp, 14
- src/functions.cpp, 3
- src/functions.hpp, 3
- src/main.cpp, 16
- src/unit\_tests.cpp, 18
- stop\_words
  - unit\_tests.cpp, 25
- stop\_words\_filename
  - unit\_tests.cpp, 27
- stop\_words\_hashes
  - unit\_tests.cpp, 25
- stop\_words\_size
  - unit\_tests.cpp, 27
- text
  - unit\_tests.cpp, 25
- text\_double\_term\_hashes
  - unit\_tests.cpp, 25
- text\_filename
  - unit\_tests.cpp, 27
- text\_single\_term\_hashes
  - unit\_tests.cpp, 26
- text\_size
  - unit\_tests.cpp, 28
- text\_triple\_term\_hashes
  - unit\_tests.cpp, 26
- unit\_tests.cpp
  - BOOST\_AUTO\_TEST\_CASE, 20–23
  - BOOST\_TEST\_MODULE, 19
  - dict\_hashes, 24
  - dict\_size, 26
  - dictionary, 24
  - dictionary\_filename, 26
  - duplicates, 26
  - duplicates\_size, 27
  - hash\_table, 24
  - hasher, 27
  - matrices, 24
  - matrix\_size, 20
  - max\_dict\_size, 20
  - max\_stop\_words\_size, 20
  - max\_term\_size, 20
  - output, 25
  - stop\_words, 25
  - stop\_words\_filename, 27
  - stop\_words\_hashes, 25
  - stop\_words\_size, 27
  - text, 25
  - text\_double\_term\_hashes, 25
  - text\_filename, 27
  - text\_single\_term\_hashes, 26
  - text\_size, 28
  - text\_triple\_term\_hashes, 26
  - words\_count, 28
- words\_count
  - unit\_tests.cpp, 28
- write\_in\_file
  - functions.hpp, 15