Corrector

0.2

Generated by Doxygen 1.8.2

Wed Dec 12 2012 15:35:48

# **Contents**

1	Orth	ograph	ic Correct	tor									1
2	Todo	List											3
3	Mod	ule Inde	ex										5
	3.1	Module	es			 	 5						
4	Data	Structi	ure Index										7
	4.1	Data S	tructures			 	 7						
5	File	Index											9
	5.1	File Lis	st			 	 9						
6	Mod	ule Doc	umentatio	on									11
	6.1	Small	chained lis	ts function	s	 	 11						
		6.1.1	Detailed	Description	n	 	 11						
		6.1.2	Function	Document	ation	 	 11						
			6.1.2.1	add_lclis	t	 	 11						
			6.1.2.2	drop_lclis	st	 	 11						
			6.1.2.3	len_lclist		 	 12						
			6.1.2.4	make_lcl	ist	 	 12						
			6.1.2.5	pop_lclis	t	 	 12						
	6.2	Correc	tion function	ons		 	 13						
		6.2.1	Detailed	Description	n	 	 13						
		6.2.2	Function	Document	ation	 	 13						
			6.2.2.1	correct .		 	 13						
			6.2.2.2	correct_a	all	 	 13						
			6.2.2.3	ten_bests	s	 	 14						
	6.3	Debug	tools			 	 15						
		6.3.1	Detailed	Description	n	 	 15						
		6.3.2	Function	Document	ation	 	 15						
			6.3.2.1	atobin .		 	 15						
			6322	hinary n	rint								15

ii CONTENTS

			6.3.2.3	itobin .				 	 	 	 	 	 15
7	Data	Structi	ure Docun	nentatio	n								17
	7.1	Ichaine	ed_list Stru	ct Refer	ence.			 	 	 	 	 	 17
		7.1.1	Detailed	Descript	ion			 	 	 	 	 	 17
	7.2	params	s Struct Re	eference				 	 	 	 	 	 17
		7.2.1	Detailed	Descript	ion			 	 	 	 	 	 18
8	File	Docum	entation										19
	8.1	inc/clis	t.h File Re	ference				 	 	 	 	 	 19
		8.1.1	Detailed	Descript	ion			 	 	 	 	 	 20
	8.2	inc/cor	rector.h Fil	le Refere	ence			 	 	 	 	 	 20
		8.2.1	Detailed	Descript	ion			 	 	 	 	 	 20
		8.2.2	Variable	Docume	ntation			 	 	 	 	 	 20
			8.2.2.1	substitu	utions .			 	 	 	 	 	 20
	8.3	inc/utils	s.h File Re	ference				 	 	 	 	 	 21
		8.3.1	Detailed	Descript	ion			 	 	 	 	 	 22
	8.4	inc/woi	rdtools.h F	ile Refer	ence .			 	 	 	 	 	 22
		8.4.1	Detailed	Descript	ion			 	 	 	 	 	 23
		8.4.2	Function	Docume	entation			 	 	 	 	 	 23
			8.4.2.1	alllen .				 	 	 	 	 	 23
			8.4.2.2	hashdid	ct_addv	vord .		 	 	 	 	 	 23
			8.4.2.3	hashdid	ct_in			 	 	 	 	 	 24
			8.4.2.4	jhash_c	char			 	 	 	 	 	 24
			8.4.2.5	levensh	ntein			 	 	 	 	 	 24
			8.4.2.6	strheq				 	 	 	 	 	 24
	8.5	src/clis	t.c File Re	ference				 	 	 	 	 	 25
		8.5.1	Detailed	Descript	ion			 	 	 	 	 	 25
	8.6	src/cor	rector.c Fil	le Refere	ence			 	 	 	 	 	 25
		8.6.1	Detailed	Descript	ion			 	 	 	 	 	 26
		8.6.2	Macro De	efinition I	Docume	entation	1	 	 	 	 	 	 26
			8.6.2.1	addtup	le			 	 	 	 	 	 26
			8.6.2.2	addtwe	ak			 	 	 	 	 	 26
			8.6.2.3	init_su	gg_threa	ad		 	 	 	 	 	 26
	8.7	src/util	s.c File Re	ference				 	 	 	 	 	 27
		8.7.1	Detailed	Descript	ion			 	 	 	 	 	 27
	8.8	src/wo	rdtools.c F	ile Refer	ence .			 	 	 	 	 	 27
		8.8.1	Detailed	Descript	ion			 	 	 	 	 	 28
		8.8.2	Function	Docume	entation			 	 	 	 	 	 28
			8.8.2.1	alllen .				 	 	 	 	 	 28
			8.8.2.2	hashdi	ct_addv	vord .		 	 	 	 	 	 28

Index			20
	8.8.2.6	strheq	. 29
	8.8.2.5	levenshtein	. 29
	8.8.2.4	jhash_char	. 28
	8.8.2.3	hashdict_in	. 28

iii

**CONTENTS** 

# **Orthographic Corrector**

#### Installation

```
make init
make
make doc
./bin/corrector ressources/dico.txt ressources/fautes2012.txt
```

## Introduction

#### Modules

Here is a quick description of the implemented modules, covering their main functionnalities.

# Levenshtein distance

· Calculate the difference between two strings, this function is utf8 friendly, and only covers utf8 specials chars.

#### Hash codes

- · Hashing a string
- · Linking each tri-gram to a list
- · Creating a trigram-dictionnary

# **Orthographic Corrector**

- · Initializing dictionnaries
- · Parsing test files
- · Correcting words
- · Comparing results

## Performances of different algorithms

## Hashing

Theese are experimental measures, hence not being in any case a valid mathematical proof.

The two compeiting implementation are a modulus (the base java hash function), and a XORing function (jhash\_char).

The idea beneath the XORing function is the following:

• If I have to sum up my function in X bits of entropy, instead of applying a modulus (even in a fast bitwise fashion), it would be way faster to "split" my checksum in sizeof(int)\*8/X block of bits and XOR them altogether

#### Scattering

- The XORing give a ±70% scattering variation.
- The modulus gives a ±150% scattering variation. The goal being a 0% scattering variation (that is to say, a perfectly uniform hash function).

Speed

The XORing runs 23% faster in average than the modulus function.

#### Collisions

Hashing a word dictionnary the XOR function produce +2% to -120% less collisions than modulus.

# **Modus Operandi**

#### **Hash Tables**

- · Create a hash table
- · Add the words related to any 3-letter tuple of the word
  - Twist some words (oe:"œ") and add them as well, with a higher ponderation

#### **Jacquard Distance**

- · Calculate the Jacquard Distance between the two words
- · If this distance is close enough to the word, proceed
- Calculate the levenshtein distance ponderated by the common tuples (this ponderation is necessary because of twisted words)

# **Todo List**

Global ten\_bests (char \*word, Iclist \*\*tuples, Iclist \*\*hashd)

Make the function Leak free

**Todo List** 

# **Module Index**

# 3.1 Modules

Here	ic a	liet c	of all	modu	، عمار

Small chained lists functions	11
Correction functions	13
Debug tools	15

6 **Module Index** 

# **Data Structure Index**

41	Data	Structu	IFAS

Here are	the data	structures	with	brief	descriptions
----------	----------	------------	------	-------	--------------

Ichained	I_list	
	A chained list structure	17
params		17

8 Data Structure Index

# File Index

# 5.1 File List

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

inc/clist.h	
Chained lists header	19
inc/corrector.h	
Word correction header	20
inc/utils.h	
Utilities header	21
inc/wordtools.h	
Word analysis utilities headers	22
src/clist.c	
Chained lists source	25
src/corrector.c	
Word corrector	25
src/utils.c	
Utilities source	27
src/wordtools.c	
Word analysis utilities source	27

10 File Index

# **Module Documentation**

# 6.1 Small chained lists functions

Minimalistic way of supporting chained lists.

#### **Functions**

```
• lclist * make_lclist ()
```

Make a new chained list.

void drop\_lclist (lclist \*I)

Drop the list.

void add\_lclist (lclist \*I, DATATYPE data)

Add an element to the list.

DATATYPE pop\_lclist (lclist \*I)

Delete the first element of the list.

int len\_lclist (lclist \*I)

Lenght of the list.

# 6.1.1 Detailed Description

Minimalistic way of supporting chained lists.

# 6.1.2 Function Documentation

```
6.1.2.1 void add_lclist ( lclist * I, DATATYPE data )
```

Add an element to the list.

# **Parameters**

1	The clist to add to
data	The data to add

Definition at line 53 of file clist.c.

6.1.2.2 void drop\_lclist ( lclist \* I )

Drop the list.

12 Module Documentation

# Parameters / The clist to drop Definition at line 43 of file clist.c. 6.1.2.3 int len\_lclist ( lclist \* / ) Lenght of the list. Parameters / The list to mesure Returns The size of the list Definition at line 78 of file clist.c.

Returns

Return an empty clist

6.1.2.4 Iclist\* make\_lclist( )

Make a new chained list.

Definition at line 32 of file clist.c.

6.1.2.5 **DATATYPE** pop\_lclist ( lclist \* I )

Delete the first element of the list.

**Parameters** 

The list to pop the last element\$

Returns

The value popped (You should free it properly if needed)

Definition at line 64 of file clist.c.

6.2 Correction functions 13

# 6.2 Correction functions

A handful correction function.

#### **Data Structures**

· struct params

## **Functions**

- void \* add\_suggestions (void \*par)
- char \*\* ten\_bests (char \*word, lclist \*\*tuples, lclist \*\*hashd)

Find the ten best correction guesses for the given word.

void correct\_all (char \*dict, char \*errs)

Correct all word in the given path.

• int correct (char \*str, char \*goal, lclist \*\*hashd, lclist \*\*tuples)

Find bests corrections for the given string.

#### **Variables**

pthread\_mutex\_t mutex [ERROR\_LIMIT]

## 6.2.1 Detailed Description

A handful correction function.

#### 6.2.2 Function Documentation

6.2.2.1 int correct ( char \* str, char \* goal, Iclist \*\* hashd, Iclist \*\* tuples )

Find bests corrections for the given string.

Correct a string.

#### **Parameters**

str	The string to look correction for
goal	The string that should be found
hashd	The hash dictionnary
tuples	The hash dictionnary of tuples

#### **Returns**

The status of the request 0: found 2: not found

Definition at line 271 of file corrector.c.

6.2.2.2 void correct\_all ( char \* dict, char \* errs )

Correct all word in the given path.

Correct all words, compare.

14 Module Documentation

## **Parameters**

dict	The dictionnary adress
errs	The error file adress
	Load the words to correct from the given path
	Create the necessary structures to proceed due correction
	Time and proceed all corrections

Definition at line 231 of file corrector.c.

6.2.2.3 char\*\* ten\_bests ( char \* word, Iclist \*\* tuples, Iclist \*\* hashd )

Find the ten best correction guesses for the given word.

Return the ten best guesses for the given word.

Todo Make the function Leak free

## **Parameters**

word	The word to correct
tuples	The tuples hashes
hashd	The dictionnary hash

# Returns

The list of the ten (or less) best guesses

Definition at line 113 of file corrector.c.

6.3 Debug tools

# 6.3 Debug tools

Simple debugging tools.

## **Functions**

void binary\_print (char \*val)

Print the given char in binary.

char \* atobin (char \*str)

Convert the char to binary format.

• char \* itobin (int nb, unsigned int size)

Convert the integer to binary format.

## 6.3.1 Detailed Description

Simple debugging tools.

## 6.3.2 Function Documentation

6.3.2.1 char\* atobin ( char \* str )

Convert the char to binary format.

#### **Parameters**

str	The string to convert to binary

#### Returns

The binary value as a string

Definition at line 39 of file utils.c.

6.3.2.2 void binary\_print ( char \* val )

Print the given char in binary.

#### **Parameters**

val	The value to print

Definition at line 31 of file utils.c.

6.3.2.3 char\* itobin ( int nb, unsigned int size )

Convert the integer to binary format.

#### **Parameters**

nb	The number to convert
size	The number of bits to print sizeof()*8

16 Module Documentation

Returns

The binary value as a string

Definition at line 57 of file utils.c.

# **Data Structure Documentation**

# 7.1 Ichained\_list Struct Reference

A chained list structure.

```
#include <clist.h>
```

## **Data Fields**

• DATATYPE data

The stored data, of type DATATYPE.

• struct lchained\_list \* next

The adress of the next list member.

# 7.1.1 Detailed Description

A chained list structure.

Definition at line 34 of file clist.h.

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

• inc/clist.h

# 7.2 params Struct Reference

## **Data Fields**

- int vMin
- int vMax
- Iclist \*\* hashd
- Iclist \*\* qualified
- unsigned int \* found\_hashs
- int \* nbmatching
- char \* word
- int max
- int MAX

# 7.2.1 Detailed Description

Definition at line 46 of file corrector.c.

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

• src/corrector.c

# **File Documentation**

# 8.1 inc/clist.h File Reference

# Chained lists header.

```
#include "utils.h"
#include <stdlib.h>
```

# **Data Structures**

struct lchained\_list

A chained list structure.

# Macros

• #define DATATYPE void\*

The type of data handled by the lists.

• #define LPOP\_ERROR NULL

Extremely important for automation.

# **Typedefs**

• typedef struct lchained\_list lclist

A chained list structure.

# **Functions**

```
lclist * make_lclist ()
```

Make a new chained list.

void drop\_lclist (lclist \*)

Drop the list.

void add\_lclist (lclist \*, DATATYPE)

Add an element to the list.

DATATYPE pop\_lclist (lclist \*)

Delete the first element of the list.

int len\_lclist (lclist \*)

Lenght of the list.

20 File Documentation

# 8.1.1 Detailed Description

Chained lists header.

Definition in file clist.h.

# 8.2 inc/corrector.h File Reference

```
Word correction header.
```

```
#include "wordtools.h"
#include <pthread.h>
```

## **Macros**

• #define MAX\_THREADS 4

Number of authorized threads, 1 means no threads.

• #define SIZE\_LIST 10

Size of the suggestion list.

• #define ERROR LIMIT 40

Percent of variation allowed towards the word.

#### **Functions**

```
char ** ten_bests (char *, lclist **, lclist **)
```

Return the ten best guesses for the given word.

void correct\_all (char \*, char \*)

Correct all words, compare.

int correct (char \*, char \*, lclist \*\*, lclist \*\*)

Correct a string.

# **Variables**

• char \* substitutions [][2] Substitutions to perform.

# 8.2.1 Detailed Description

Word correction header.

Definition in file corrector.h.

# 8.2.2 Variable Documentation

#### 8.2.2.1 char\* substitutions[][2]

## Initial value:

```
={
{ "oe", "œ"}
}
```

Substitutions to perform.

Definition at line 36 of file corrector.h.

# 8.3 inc/utils.h File Reference

#### Utilities header.

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

## **Macros**

• #define WHERE printf("In %s line %d (%s)\n",\_\_FILE\_\_,\_LINE\_\_,\_func\_\_)

Print the current position on the program.

#define ERROR(msg) {WHERE; perror(msg);printf("\n");exit(EXIT\_FAILURE);}

Print the current error, with the error message msg and exits.

#define OUT(msg) {WHERE; fprintf(stderr,msg);fprintf(stderr,"\n");exit(EXIT\_FAILURE);}

Exits the program with the message msg.

• #define FPRINT(msg) {fprintf(stdout,msg);fflush(stdout);}

Print msg in stdout and flush.

• #define min(a, b) ((a)<(b)?(a):(b))

Return the max beetween the two parameters.

• #define TIMER\_INIT struct timeval tvBegin, tvEnd

Initialise the timer.

• #define TIMER\_STRT gettimeofday(&tvBegin, NULL)

Start the timer.

• #define TIMER\_STOP gettimeofday(&tvEnd, NULL)

Stop the timer.

#define TIMER\_USEC ((tvEnd.tv\_usec+1000000\*tvEnd.tv\_sec)-(tvBegin.tv\_usec+1000000\*tvBegin.tv\_sec))

Get the time interval.

#### **Functions**

void binary\_print (char \*)

Print the given char in binary.

• char \* itobin (int, unsigned int)

Convert the integer to binary format.

char \* atobin (char \*)

Convert the char to binary format.

#### **Variables**

· int verbose

Verbosity of the program.

22 File Documentation

# 8.3.1 Detailed Description

Utilities header. All utilities are defined and documented here

Definition in file utils.h.

#### 8.4 inc/wordtools.h File Reference

#### Word analysis utilities headers.

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <unistd.h>
#include "utils.h"
#include "utf8.h"
#include "clist.h"
```

#### **Macros**

```
• #define min3(a, b, c) min(min((a),(b)),(c))
```

Minimal value between three.

• #define HASH\_S\_SIZE 16

Size of the checksum (in bits)

• #define HASH\_M\_SIZE 20

Size of the checksum (in bits)

#define HASH\_L\_SIZE 24

Size of the checksum (in bits)

#define HASH\_SIZE HASH\_M\_SIZE

Dictionnary standard size.

• #define HASH\_S\_DSIZ 65536

Dictionnary size (2\*\*HASH\_SIZE)

#define HASH\_M\_DSIZ 1048576

Dictionnary size (2\*\*HASH\_SIZE)

#define HASH\_L\_DSIZ 16777216

Dictionnary size (2\*\*HASH\_SIZE)

#define HASH\_DSIZ HASH\_M\_DSIZ

Dictionnary standard size.

#define jhash(x) jhash\_char((x),HASH\_SIZE)

Link to the hash function, for flexibility purpose.

#define jhash\_S(x) jhash\_char((x),HASH\_S\_SIZE)

Small sized hash.

#define jhash\_M(x) jhash\_char((x),HASH\_M\_SIZE)

Medium sized hash.

#define jhash\_L(x) jhash\_char((x),HASH\_M\_SIZE)

Large sized hash.

• #define HASH\_POWR 5

Power of two to elevate the checksum.

#define FRMT\_DICT "%[^\n]\n"

Format of the dictionnary to retreive.

#define FRMT\_ERRS "%[^>]>%[^\n]\n"

Format of the corrections to make.

## **Functions**

unsigned int levenshtein (char \*, char \*, int, int)

The levenshtein function Note that the function return the ponderated levenshtein function with 2 digits precision.

unsigned int jhash\_char (const char \*, int)

Hash the given word according to the java string hash function.

lclist \*\* build\_hashdict (char \*)

Build the hash dictionnary of the file at the given path.

void hashdict\_addword (Iclist \*\*, unsigned int, char \*, int, unsigned int)

Add a word to a dictionnary.

int hashdict\_in (lclist \*\*, char \*)

Check if the string is in the hashed values.

lclist \*\* build\_3tupledict (char \*)

Build the 3-tuple dictionnary.

int strheq (const char \*, const char \*)

Compare the jhash function between two strings.

void allen (char \*, int \*, int \*)

Calculate both lenghts for speed purpose.

## 8.4.1 Detailed Description

Word analysis utilities headers.

Definition in file wordtools.h.

## 8.4.2 Function Documentation

8.4.2.1 void allen ( char \* word, int \* ascii, int \* utf )

Calculate both lenghts for speed purpose.

#### **Parameters**

word	The word to measure
ascii	The ascii length
utf	The utf length

Definition at line 183 of file wordtools.c.

8.4.2.2 void hashdict\_addword ( Iclist \*\* hashd, unsigned int hash, char \* str, int subhash, unsigned int max )

Add a word to a dictionnary.

# **Parameters**

hashd	The hash dictionnary (a chained list)
hash	The hash of the given object
str	The object hashed, to store if unique
subhash	Check if collisions are genuine
max	fashion, in the way that is doesn't calculate the hash and solve the collision, but relie on the
	calling function to provide the tools for it.

Definition at line 132 of file wordtools.c.

24 File Documentation

8.4.2.3 int hashdict\_in ( Iclist \*\* hashd, char \* str )

Check if the string is in the hashed values.

#### **Parameters**

hashd	The hash dictionnary
str	The string to look for in the dictionnary

#### Returns

true if the word is in false otherwise

Definition at line 162 of file wordtools.c.

8.4.2.4 unsigned int jhash\_char ( const char \* word, int hash\_size )

Hash the given word according to the java string hash function.

#### **Parameters**

ĺ	word	The word to hash
	hash_size	Size of the hash (in bits)

#### Returns

The unsigned int of the hash

Definition at line 59 of file wordtools.c.

8.4.2.5 unsigned int levenshtein ( char \* w1, char \* w2, int l, int l2 )

The levenshtein function Note that the function return the ponderated levenshtein function with 2 digits precision.

#### **Parameters**

w1,w2	The words to compare
1,12	The length of the words to compare

#### Returns

the distance, in permutation and additions/deletion between the two words aka the levenshtein distance

Definition at line 29 of file wordtools.c.

8.4.2.6 int strheq ( const char \* w1, const char \* w2 )

Compare the jhash function between two strings.

#### **Parameters**

w1,w2	The words to compare

#### Returns

1 if differents 0 if equal (for conformity purpose with strcmp)

Definition at line 174 of file wordtools.c.

## 8.5 src/clist.c File Reference

```
Chained lists source.
```

```
#include "clist.h"
```

## **Functions**

lclist \* make\_lclist ()

Make a new chained list.

void drop\_lclist (lclist \*I)

Drop the list.

void add\_lclist (lclist \*I, DATATYPE data)

Add an element to the list.

DATATYPE pop\_lclist (lclist \*I)

Delete the first element of the list.

• int len\_lclist (lclist \*I)

Lenght of the list.

# 8.5.1 Detailed Description

Chained lists source.

Definition in file clist.c.

# 8.6 src/corrector.c File Reference

## Word corrector.

```
#include "corrector.h"
```

# **Data Structures**

struct params

# **Macros**

- #define init\_sugg\_thread(i, j)
- #define **newW** ((char\*)node->data)
- #define t(i) tuple[i]
- #define w(j) news[i+j]
- #define **S**(j) substitutions[0][1][j]
- #define **s**(j) substitutions[0][0][j]
- #define addtuple
- #define addtweak

26 File Documentation

# **Functions**

```
void * add_suggestions (void *par)
```

char \*\* ten\_bests (char \*word, lclist \*\*tuples, lclist \*\*hashd)

Find the ten best correction guesses for the given word.

void correct\_all (char \*dict, char \*errs)

Correct all word in the given path.

• int correct (char \*str, char \*goal, lclist \*\*hashd, lclist \*\*tuples)

Find bests corrections for the given string.

int main (int argc, char \*argv[])

Process correction over the given parameters.

#### **Variables**

pthread\_mutex\_t mutex [ERROR\_LIMIT]

# 8.6.1 Detailed Description

Word corrector.

Definition in file corrector.c.

#### 8.6.2 Macro Definition Documentation

#### 8.6.2.1 #define addtuple

#### Value:

#### 8.6.2.2 #define addtweak

## Value:

# 8.6.2.3 #define init\_sugg\_thread( i, j)

#### Value:

```
P->vMin=i;\
P->vMax=j;\
P->hashd=hashd;

qualified;\
p->qualified=
found_hashs;\
nbmatching;\
P->nbmatching=

P->word=word;\
P->max=max;\
P->MAX=MAX;
```

Definition at line 29 of file corrector.c.

#### 8.7 src/utils.c File Reference

#### Utilities source.

```
#include "utils.h"
```

## **Functions**

void binary\_print (char \*val)

Print the given char in binary.

char \* atobin (char \*str)

Convert the char to binary format.

• char \* itobin (int nb, unsigned int size)

Convert the integer to binary format.

#### 8.7.1 Detailed Description

Utilities source.

Definition in file utils.c.

# 8.8 src/wordtools.c File Reference

Word analysis utilities source.

```
#include "wordtools.h"
```

#### Macros

#define GT(t, i, j) t[(i)+(j)\*(l)]
 Acess to element t[i,j].

#### **Functions**

• unsigned int levenshtein (char \*w1, char \*w2, int I, int I2)

The levenshtein function Note that the function return the ponderated levenshtein function with 2 digits precision.

• unsigned int jhash\_char (const char \*word, int hash\_size)

Hash the given word according to the java string hash function.

lclist \*\* build\_hashdict (char \*path)

Build the hash dictionnary of the file at the given path.

lclist \*\* build\_3tupledict (char \*path)

Build the 3-tuple dictionnary.

void hashdict\_addword (Iclist \*\*hashd, unsigned int hash, char \*str, int subhash, unsigned int max)

Add a word to a dictionnary.

int hashdict\_in (lclist \*\*hashd, char \*str)

Check if the string is in the hashed values.

• int strheq (const char \*w1, const char \*w2)

Compare the jhash function between two strings.

void allen (char \*word, int \*ascii, int \*utf)

Calculate both lenghts for speed purpose.

28 File Documentation

# 8.8.1 Detailed Description

Word analysis utilities source.

Definition in file wordtools.c.

#### 8.8.2 Function Documentation

8.8.2.1 void allen ( char \* word, int \* ascii, int \* utf )

Calculate both lenghts for speed purpose.

#### **Parameters**

word	The word to measure
ascii	The ascii length
utf	The utf length

Definition at line 183 of file wordtools.c.

8.8.2.2 void hashdict\_addword ( lclist \*\* hashd, unsigned int hash, char \* str, int subhash, unsigned int max )

Add a word to a dictionnary.

#### **Parameters**

hashd	The hash dictionnary (a chained list)
hash	The hash of the given object
str	The object hashed, to store if unique
subhash	Check if collisions are genuine
max	The lenght of the word (hence the number of 3-tuples) This function works in a very particular fashion, in the way that is doesn't calculate the hash and solve the collision, but relie on the calling function to provide the tools for it.

Definition at line 132 of file wordtools.c.

8.8.2.3 int hashdict\_in ( Iclist \*\* hashd, char \* str )

Check if the string is in the hashed values.

#### **Parameters**

hashd	The hash dictionnary
str	The string to look for in the dictionnary

#### Returns

true if the word is in false otherwise

Definition at line 162 of file wordtools.c.

8.8.2.4 unsigned int jhash\_char ( const char \* word, int hash\_size )

Hash the given word according to the java string hash function.

#### **Parameters**

word	The word to hash
hash_size	Size of the hash (in bits)

#### **Returns**

The unsigned int of the hash

Definition at line 59 of file wordtools.c.

8.8.2.5 unsigned int levenshtein ( char \* w1, char \* w2, int I, int I2 )

The levenshtein function Note that the function return the ponderated levenshtein function with 2 digits precision.

#### **Parameters**

w1,w2	The words to compare
1,12	The length of the words to compare

#### **Returns**

the distance, in permutation and additions/deletion between the two words aka the levenshtein distance

Definition at line 29 of file wordtools.c.

8.8.2.6 int strheq ( const char \* w1, const char \* w2 )

Compare the jhash function between two strings.

# Parameters

w1,w2	The words to compare

#### Returns

1 if differents 0 if equal (for conformity purpose with strcmp)

Definition at line 174 of file wordtools.c.

# Index

add_lclist	Debug tools, 15
Small chained lists functions, 11	
addtuple	jhash_char
corrector.c, 26	wordtools.c, 28
addtweak	wordtools.h, 24
corrector.c, 26	
alllen	Ichained_list, 17
wordtools.c, 28	len_lclist
wordtools.h, 23	Small chained lists functions, 12
atobin	levenshtein
Debug tools, 15	wordtools.c, 29
	wordtools.h, 24
binary_print	
Debug tools, 15	make_lclist
	Small chained lists functions, 12
correct	noromo 17
Correction functions, 13	params, 17
correct_all	pop_lclist
Correction functions, 13	Small chained lists functions, 12
Correction functions, 13	Small chained lists functions, 11
correct, 13	add_lclist, 11
correct_all, 13	drop Iclist, 11
ten_bests, 14	. —
corrector.c	len_lclist, 12
addtuple, 26	make_lclist, 12
addtweak, 26	pop_lclist, 12
init_sugg_thread, 26	src/clist.c, 25
corrector.h	src/corrector.c, 25
substitutions, 20	src/utils.c, 27
	src/wordtools.c, 27
Debug tools, 15	strheq
atobin, 15	wordtools.c, 29
binary_print, 15	wordtools.h, 24
itobin, 15	substitutions
drop_lclist	corrector.h, 20
Small chained lists functions, 11	ton boots
landa disk on delivered	ten_bests
hashdict_addword	Correction functions, 14
wordtools.c, 28	wordtools.c
wordtools.h, 23	allen, 28
hashdict_in	hashdict_addword, 28
wordtools.c, 28	hashdict in, 28
wordtools.h, 23	jhash_char, 28
ing/gligt h 10	· —
inc/clist.h, 19	levenshtein, 29
inc/corrector.h, 20	strheq, 29
inc/utils.h, 21	wordtools.h
inc/wordtools.h, 22	allen, 23
init_sugg_thread	hashdict_addword, 23
corrector.c, 26	hashdict_in, 23
itobin	jhash_char, 24

INDEX 31

levenshtein, 24 strheq, 24