Corrector

0.2

Generated by Doxygen 1.8.2

Wed Dec 12 2012 15:35:48

Contents

Chapter 1

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)

Chapter 2

Todo List

Global ten_bests (char ∗word, lclist ∗∗tuples, lclist ∗∗hashd)

Make the function Leak free

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

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

Chapter 6

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 ∗l)

  Drop the list.
- void add_lclist (lclist ∗l, DATATYPE data)

  Add an element to the list.
- DATATYPE pop_lclist (lclist ∗l)

  Delete the first element of the list.
- int len_lclist (lclist ∗l)

  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** ∗ *l*, **DATATYPE** *data*)

Add an element to the list.

Parameters

l	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** ∗ *l*)

Drop the list.

Parameters

l	The clist to drop

Definition at line 43 of file clist.c.

6.1.2.3 int len_lclist (**lclist** ∗ *l*)

Lenght of the list.

Parameters

l	The list to mesure

Returns

The size of the list

Definition at line 78 of file clist.c.

6.1.2.4 **lclist**∗ make_lclist ( )

Make a new chained list.

Returns

Return an empty clist

Definition at line 32 of file clist.c.

6.1.2.5 **DATATYPE** pop_lclist (**lclist** ∗ *l*)

Delete the first element of the list.

Parameters

l	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

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,* lclist ∗∗ *hashd,* lclist ∗∗ *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.

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,* lclist ∗∗ *tuples,* lclist ∗∗ *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

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

Returns

The binary value as a string

Definition at line 57 of file utils.c.

Chapter 7

Data Structure Documentation

7.1 lchained_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**
- lclist ∗∗ **hashd**
- lclist ∗∗ **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

Chapter 8

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.

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.

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 (lclist ∗∗, 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 alllen (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 alllen (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 (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.

8.4.2.3 int hashdict_in (lclist ** *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

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
l,l2	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 ∗l)

  Drop the list.
- void add_lclist (lclist ∗l, DATATYPE data)

  Add an element to the list.
- DATATYPE pop_lclist (lclist ∗l)

  Delete the first element of the list.
- int len_lclist (lclist ∗l)

  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**

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:

```
hash=jhash(tuple);\
                                                              suggests[k++]=tuples[hash];
```

8.6.2.2 #define addtweak

Value:

```
hash=jhash(tuple);\
                                                              tweaks[l++]=tuples[hash];
```

8.6.2.3 #define init_sugg_thread(*i*, *j*)

Value:

```
P->vMin=i;\
                                                                                 P->vMax=j;\
                                                                                 P->hashd=hashd;
        \
                                                                                 P->qualified=
        qualified;\
                                                                                 P->found_hashs=
        found_hashs;\
                                                                                 P->nbmatching=
        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 l, int l2)

  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 (lclist ∗∗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 alllen (char ∗word, int ∗ascii, int ∗utf)

  Calculate both lenghts for speed purpose.

8.8.1 Detailed Description

Word analysis utilities source.

Definition in file wordtools.c.

8.8.2 Function Documentation

8.8.2.1 void alllen (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 (lclist ∗∗ *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 *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
l,l2	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