PLSA (multiprocessor)

1.0.1

Generated by Doxygen 1.5.6

Sun Mar 7 16:16:26 2010

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 cooccur Struct Reference

`#include <plsa-defn.h>`

Data Fields

- PROBNODE x
- unsigned int column

3.1.1 Field Documentation

3.1.1.1 PROBNODE cooccur::x

The co-occurrence count, as a log value

3.1.1.2 unsigned int cooccur::column

Column position of this value

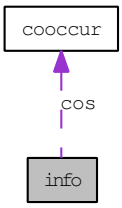
The documentation for this struct was generated from the following file:

- plsa-defn.h

3.2 info Struct Reference

`#include <plsa-defn.h>`

Collaboration diagram for info:



Data Fields

- bool verbose
- bool debug
- bool textio
- bool rounding
- bool no_output
- unsigned int seed
- unsigned int num_clusters
- char * base_fn
- unsigned int maxiter
- unsigned int snapshot
- unsigned int m
- unsigned int n
- char * co_fn
- COOCCUR ** cos
- unsigned int * row_ids
- unsigned int * column_ids
- unsigned int iter
- PROBNODE * probw1_z_curr
- PROBNODE * probw2_z_curr
- PROBNODE * probz_curr
- PROBNODE * probw1_z_prev
- PROBNODE * probw2_z_prev
- PROBNODE * probz_prev
- PROBNODE * prob_w1w2
- int threads
- signed int world_id
- signed int world_size
- unsigned int block_start
- unsigned int block_end
- unsigned int block_size
- unsigned int sigfpe_count
- time_t program_start
- double run_time
- double readCO_time

- double initEM_time
- double calculateProbW1W2_time
- double calculateML_time
- double swapPrevCurr_time
- double applyEMStep_time
- double gatherProbs_time
- double normalizeProbs_time
- double distributeProbs_time
- double printCoProbs_time
- time_t program_end

3.2.1 Field Documentation

3.2.1.1 bool info::verbose

Verbose output?

3.2.1.2 bool info::debug

Debugging output?

3.2.1.3 bool info::textio

Text I/O

3.2.1.4 bool info::rounding

Should the output values be rounded?

3.2.1.5 bool info::no_output

Suppress output

3.2.1.6 unsigned int info::seed

Random seed

3.2.1.7 unsigned int info::num_clusters

Number of clusters

3.2.1.8 char* info::base_fn

Base filename for the output file

3.2.1.9 unsigned int info::maxiter

Maximum number of iterations

3.2.1.10 unsigned int info::snapshot

Intervals to output p(x,y); UINT_MAX means do not output

3.2.1.11 unsigned int info::m

Number of unique query terms

3.2.1.12 unsigned int info::n

Number of terms in the document collection

3.2.1.13 char* info::co_fn

Co-occurrence filename

3.2.1.14 COOCCUR** info::cos

Co-occurrence counts in a COOCCUR data structure

3.2.1.15 unsigned int* info::row_ids

List of row identifiers (m of them)

3.2.1.16 unsigned int* info::column_ids

List of column identifiers (m of them)

3.2.1.17 unsigned int info::iter

Iteration; only calculated by the main process and broadcasted to others

3.2.1.18 PROBNODE* info::probw1_z_curr

P(w1|z) of size (k * m)

3.2.1.19 PROBNODE* info::probw2_z_curr

P(w2|z) of size (k * n)

3.2.1.20 PROBNODE* info::probz_curr

P(z) of size (k)

3.2.1.21 PROBNODE* info::probw1_z_prev

P'(w1|z) of size (k * m)

3.2.1.22 PROBNODE* info::probw2_z_prev

P'(w2|z) of size (k * n)

3.2.1.23 PROBNODE* info::probz_prev

P'(z) of size (k)

3.2.1.24 PROBNODE* info::prob_w1w2

P(w1,w2) of size (m * n)

3.2.1.25 int info::threads

3.2.1.26 signed int info::world_id

ID of this process

3.2.1.27 signed int info::world_size

Number of processes total

3.2.1.28 unsigned int info::block_start

Starting block (cluster) for this process to handle

3.2.1.29 unsigned int info::block_end

Ending block (cluster) for this process to handle

3.2.1.30 unsigned int info::block_size

Size of the block for this process to handle

3.2.1.31 unsigned int info::sigfpe_count

Number of floating point exception errors

3.2.1.32 time_t info::program_start

3.2.1.33 double info::run_time

3.2.1.34 double info::readCO_time

3.2.1.35 double info::initEM_time

3.2.1.36 double info::calculateProbW1W2_time

3.2.1.37 double info::calculateML_time

3.2.1.38 double info::swapPrevCurr_time

3.2.1.39 double info::applyEMStep_time

3.2.1.40 double info::gatherProbs_time

3.2.1.41 double info::normalizeProbs_time

3.2.1.42 double info::distributeProbs_time

3.2.1.43 double info::printCoProbs_time

3.2.1.44 time_t info::program_end

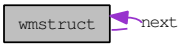
The documentation for this struct was generated from the following file:

- plsa-defn.h

3.3 wmstruct Struct Reference

`#include <wmalloc.h>`

Collaboration diagram for wmstruct:



Data Fields

- void * ptr
- size_t size
- char * file
- unsigned int line
- struct wmstruct * next

3.3.1 Field Documentation

3.3.1.1 void* wmstruct::ptr

3.3.1.2 size_t wmstruct::size

3.3.1.3 char* wmstruct::file

3.3.1.4 unsigned int wmstruct::line

3.3.1.5 struct wmstruct* wmstruct::next [read]

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

- wmalloc.h

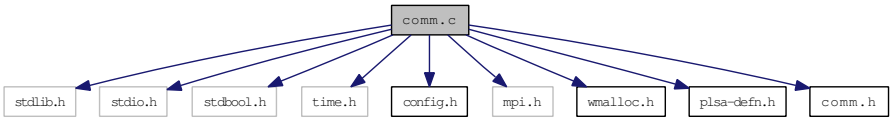
Chapter 4

File Documentation

4.1 comm.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <time.h>
#include "config.h"
#include <mpi.h>
#include "wmalloc.h"
#include "plsa-defn.h"
#include "comm.h"
```

Include dependency graph for comm.c:



Functions

- static void recvProbsFromOthers (INFO *info)
- static void sendProbsToMain (INFO *info)
- static void sendProbsToOthers (INFO *info)
- static void recvProbsFromMain (INFO *info)
- void distributeProbs (INFO *info)
- void gatherProbs (INFO *info)

4.1.1 Function Documentation

4.1.1.1 void distributeProbs (INFO ∗ *info*)

MAINPROC sends the initialized (∗current∗) p(i|z), p(j|z), and p(z) to all other processes

4.1.1.2 void gatherProbs (INFO ∗ *info*)

All other processes send the ∗current∗ p(i|z), p(j|z), and p(z) to MAINPROC

4.1.1.3 static void recvProbsFromMain (INFO ∗ *info*) [static]

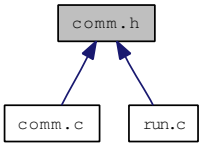
4.1.1.4 static void recvProbsFromOthers (INFO ∗ *info*) [static]

4.1.1.5 static void sendProbsToMain (INFO ∗ *info*) [static]

4.1.1.6 static void sendProbsToOthers (INFO ∗ *info*) [static]

4.2 comm.h File Reference

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



Functions

- void distributeProbs (INFO ∗info)
- void gatherProbs (INFO ∗info)

4.2.1 Function Documentation

4.2.1.1 void distributeProbs (INFO ∗ *info*)

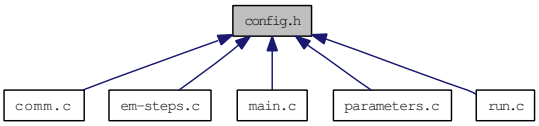
MAINPROC sends the initialized (∗current∗) p(i|z), p(j|z), and p(z) to all other processes

4.2.1.2 void gatherProbs (INFO ∗ *info*)

All other processes send the ∗current∗ p(i|z), p(j|z), and p(z) to MAINPROC

4.3 config.h File Reference

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



Defines

- #define HAVE_FLOAT_H 1
- #define HAVE_INTTYPES_H 1
- #define HAVE_LIMITS_H 1
- #define HAVE_MALLOC 1
- #define HAVE_MEMORY_H 1
- #define HAVE_MPI 1
- #define HAVE_OPENMP 1
- #define HAVE_REALLOC 1
- #define HAVE_STDBOOL_H 1
- #define HAVE_STDINT_H 1
- #define HAVE_STDLIB_H 1
- #define HAVE_STRINGS_H 1
- #define HAVE_STRING_H 1
- #define HAVE_SYS_STAT_H 1
- #define HAVE_SYS_TYPES_H 1
- #define HAVE_UNISTD_H 1
- #define HAVE__BOOL 1
- #define PACKAGE "plsa-mp"
- #define PACKAGE_BUGREPORT "r.wan@aist.go.jp"
- #define PACKAGE_NAME "PLSA-MP"
- #define PACKAGE_STRING "PLSA-MP 1.0"
- #define PACKAGE_TARNAME "plsa-mp"
- #define PACKAGE_VERSION "1.0"
- #define STDC_HEADERS 1
- #define VERSION "1.0"

4.3.1 Define Documentation

4.3.1.1 #define HAVE__BOOL 1

4.3.1.2 #define HAVE_FLOAT_H 1

4.3.1.3 #define HAVE_INTTYPES_H 1

4.3.1.4 #define HAVE_LIMITS_H 1

4.3.1.5 #define HAVE_MALLOC 1

4.3.1.6 #define HAVE_MEMORY_H 1

4.3.1.7 #define HAVE_MPI 1

4.3.1.8 #define HAVE_OPENMP 1

4.3.1.9 #define HAVE_REALLOC 1

4.3.1.10 #define HAVE_STDBOOL_H 1

4.3.1.11 #define HAVE_STDINT_H 1

4.3.1.12 #define HAVE_STDLIB_H 1

4.3.1.13 #define HAVE_STRING_H 1

4.3.1.14 #define HAVE_STRINGS_H 1

4.3.1.15 #define HAVE_SYS_STAT_H 1

4.3.1.16 #define HAVE_SYS_TYPES_H 1

4.3.1.17 #define HAVE_UNISTD_H 1

4.3.1.18 #define PACKAGE "plsa-mp"

4.3.1.19 #define PACKAGE_BUGREPORT "r.wan@aist.go.jp"

4.3.1.20 #define PACKAGE_NAME "PLSA-MP"

4.3.1.21 #define PACKAGE_STRING "PLSA-MP 1.0"

4.3.1.22 #define PACKAGE_TARNAME "plsa-mp"

4.3.1.23 #define PACKAGE_VERSION "1.0"

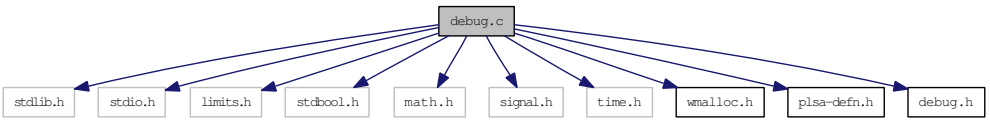
4.3.1.24 #define STDC_HEADERS 1

4.3.1.25 #define VERSION "1.0"

4.4 debug.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>
#include <math.h>
#include <signal.h>
#include <time.h>
#include "wmalloc.h"
#include "plsa-defn.h"
#include "debug.h"
```

Include dependency graph for debug.c:



Functions

- void handler_sigfpe ()
- void debugCheckCo (INFO *info)
- void checkCoProb (INFO *info)
- void printAllProbsPrev (INFO *info)
- void printAllProbsCurr (INFO *info)
- void printJointProb (INFO *info)

4.4.1 Function Documentation

4.4.1.1 void checkCoProb (INFO * *info*)

4.4.1.2 void debugCheckCo (INFO * *info*)

4.4.1.3 void handler_sigfpe ()

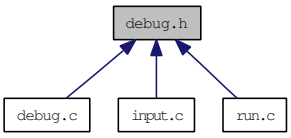
4.4.1.4 void printAllProbsCurr (INFO * *info*)

4.4.1.5 void printAllProbsPrev (INFO * *info*)

4.4.1.6 void printJointProb (INFO * *info*)

4.5 debug.h File Reference

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



Functions

- void handler_sigfpe ()
- void debugCheckCo (INFO *info)
- void checkCoProb (INFO *info)
- void printAllProbsPrev (INFO *info)
- void printAllProbsCurr (INFO *info)
- void printJointProb (INFO *info)

4.5.1 Function Documentation

4.5.1.1 void checkCoProb (INFO * *info*)

4.5.1.2 void debugCheckCo (INFO * *info*)

4.5.1.3 void handler_sigfpe ()

4.5.1.4 void printAllProbsCurr (INFO * *info*)

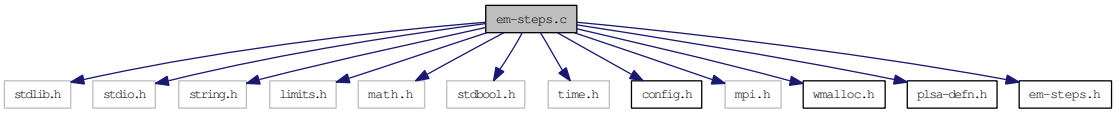
4.5.1.5 void printAllProbsPrev (INFO * *info*)

4.5.1.6 void printJointProb (INFO * *info*)

4.6 em-steps.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <limits.h>
#include <math.h>
#include <stdbool.h>
#include <time.h>
#include "config.h"
#include <mpi.h>
#include "wmalloc.h"
#include "plsa-defn.h"
#include "em-steps.h"
```

Include dependency graph for em-steps.c:



Functions

- void swapPrevCurr (INFO *info)
- void initEM (INFO *info)
- void applyEMStep (INFO *info)
- PROBNODE calculateML (INFO *info)
- void calculateProbW1W2 (INFO *info)
- void normalizeProbs (INFO *info)

4.6.1 Function Documentation

4.6.1.1 void applyEMStep (INFO * *info*)

4.6.1.2 PROBNODE calculateML (INFO * *info*)

4.6.1.3 void calculateProbW1W2 (INFO * *info*)

4.6.1.4 void initEM (INFO * *info*)

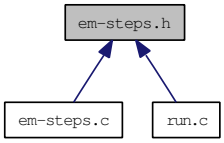
4.6.1.5 void normalizeProbs (INFO * *info*)

Normalize probabilities

4.6.1.6 void swapPrevCurr (INFO * *info*)

4.7 em-steps.h File Reference

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



Functions

- void swapPrevCurr (INFO *info)
- void initEM (INFO *info)
- void applyEMStep (INFO *info)
- PROBNODE calculateML (INFO *info)
- void calculateProbW1W2 (INFO *info)
- void normalizeProbs (INFO *info)

4.7.1 Function Documentation

4.7.1.1 void applyEMStep (INFO * *info*)

4.7.1.2 PROBNODE calculateML (INFO * *info*)

4.7.1.3 void calculateProbW1W2 (INFO * *info*)

4.7.1.4 void initEM (INFO * *info*)

4.7.1.5 void normalizeProbs (INFO * *info*)

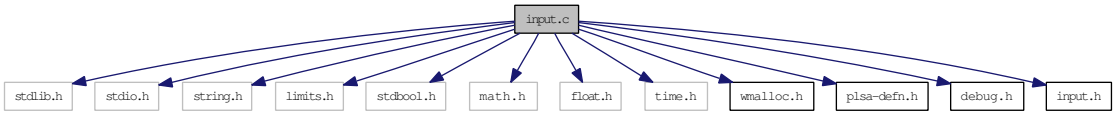
Normalize probabilities

4.7.1.6 void swapPrevCurr (INFO * *info*)

4.8 input.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <limits.h>
#include <stdbool.h>
#include <math.h>
#include <float.h>
#include <time.h>
#include "wmalloc.h"
#include "plsa-defn.h"
#include "debug.h"
#include "input.h"
```

Include dependency graph for input.c:



Functions

- void initializePostInput (INFO *info)
- bool readCO (INFO *info)

4.8.1 Function Documentation

4.8.1.1 void initializePostInput (INFO * *info*)

Initialization that depends on the input file or parameters

4.8.1.2 bool readCO (INFO * *info*)

Read the co-occurrence data from file. The format of the file is:

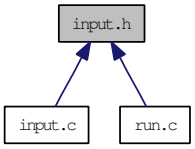
[rows][columns][row id+][column id+][w1 cos_count (w21 c21) ... (w2n c2n)]+**

row and column ids are integer values that map to the original vocabulary. The number of values should be (info -> m) and (info -> n), respectively.

Every value is an unsigned integer in binary format, unless textmode is TRUE – if so, values are in text, separated by white space (tab).

4.9 input.h File Reference

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



Functions

- void initializePostInput (INFO *info)
- bool readCO (INFO *info)

4.9.1 Function Documentation

4.9.1.1 void initializePostInput (INFO * *info*)

Initialization that depends on the input file or parameters

4.9.1.2 bool readCO (INFO * *info*)

Read the co-occurrence data from file. The format of the file is:

[rows][columns][row id+][column id+][w1 cos_count (w21 c21) ... (w2n c2n)]+**

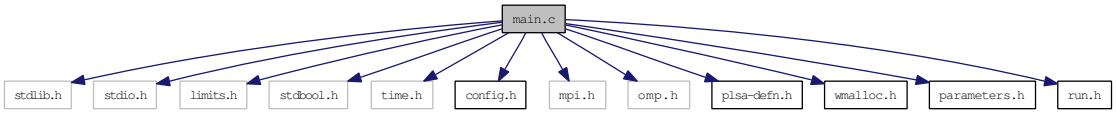
row and column ids are integer values that map to the original vocabulary. The number of values should be (info -> m) and (info -> n), respectively.

Every value is an unsigned integer in binary format, unless textmode is TRUE – if so, values are in text, separated by white space (tab).

4.10 main.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>
#include <time.h>
#include "config.h"
#include <mpi.h>
#include <omp.h>
#include "plsa-defn.h"
#include "wmalloc.h"
#include "parameters.h"
#include "run.h"
```

Include dependency graph for main.c:



Functions

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

4.10.1 Function Documentation

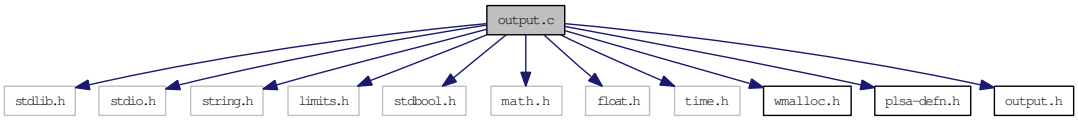
4.10.1.1 int main (int *argc*, char * *argv*[ ])

Main function

4.11 output.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <limits.h>
#include <stdbool.h>
#include <math.h>
#include <float.h>
#include <time.h>
#include "wmalloc.h"
#include "plsa-defn.h"
#include "output.h"
```

Include dependency graph for output.c:



Functions

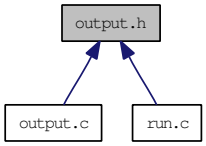
- void printCoProb (INFO *info)

4.11.1 Function Documentation

4.11.1.1 void printCoProb (INFO * *info*)

4.12 output.h File Reference

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



Functions

- void printCoProb (INFO *info)

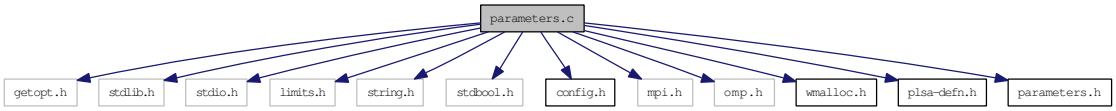
4.12.1 Function Documentation

4.12.1.1 void printCoProb (INFO * *info*)

4.13 parameters.c File Reference

```
#include <getopt.h>
#include <stdlib.h>
#include <stdio.h>
#include <limits.h>
#include <string.h>
#include <stdbool.h>
#include "config.h"
#include <mpi.h>
#include <omp.h>
#include "wmalloc.h"
#include "plsa-defn.h"
#include "parameters.h"
```

Include dependency graph for parameters.c:



Defines

- #define _GNU_SOURCE

Functions

- void usage (char *progname)
- bool checkSettings (INFO *info)
- bool processOptions (int argc, char *argv[ ], INFO *info)

4.13.1 Define Documentation

4.13.1.1 #define _GNU_SOURCE

4.13.2 Function Documentation

4.13.2.1 bool checkSettings (INFO * *info*)

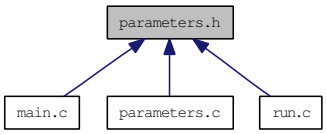
4.13.2.2 bool processOptions (int *argc*, char * *argv*[ ], INFO * *info*)

4.13.2.3 void usage (char * *progname*)

Print out usage information

4.14 parameters.h File Reference

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



Functions

- void usage (char *progname)
- bool checkSettings (INFO *info)
- bool processOptions (int argc, char *argv[ ], INFO *info)

4.14.1 Function Documentation

4.14.1.1 bool checkSettings (INFO * *info*)

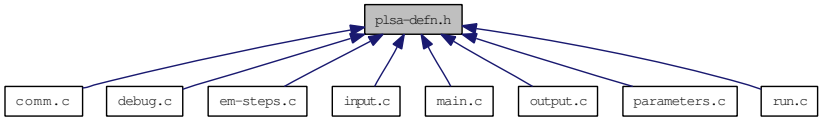
4.14.1.2 bool processOptions (int *argc*, char * *argv*[ ], INFO * *info*)

4.14.1.3 void usage (char * *progname*)

Print out usage information

4.15 plsa-defn.h File Reference

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



Data Structures

- struct cooccur
- struct info

Defines

- #define MPI_TYPE MPI_DOUBLE
- #define LN_LIMIT 23.02585093
- #define MIN_PROB (1.0E-24)
- #define DOLOG(X) (logf (X))
- #define DOEXP(X) (expf (X))
- #define DOLOGONE(X) (log1pf (X))
- #define DOLOG1PEXP(x) DOLOGONE(DOEXP(x))
- #define RANDOM_FLOAT ((PROBNODE)rand () / ((PROBNODE)RAND_MAX + (PROBNODE)1.0))
- #define DBL_LESS(A, B) ((B - A) > DBL_EPSILON)
- #define ML_DELTA 0.001
- #define MAINPROC 0
- #define ROUND_DIGITS 100000000
- #define BLOCK_LOW(id, p, n) ((id) ∗ (n)/(p))
- #define BLOCK_HIGH(id, p, n) (BLOCK_LOW ((id)+1, p, n) - 1)
- #define BLOCK_SIZE(id, p, n) (BLOCK_LOW ((id) + 1, p, n)-BLOCK_LOW(id, p, n))
- #define BLOCK_OWNER(index, p, n) (((p) ∗ ((index)+1)-1)/(n))
- #define MAX_CLUSTERS 1000
- #define MSG_TAG(X, Y, Z) ((X ∗ 10000) + (Y ∗ MAX_CLUSTERS) + Z)
- #define TAG_PROBW1_Z 1
- #define TAG_PROBW2_Z 2
- #define TAG_PROBZ 3
- #define TAG_PROBZ_W1W2 4
- #define TAG_PROBW1W2 5
- #define MSG_SEND_STATUS(V, W, X, Y, Z) ;
- #define MSG_RECV_STATUS(V, W, X, Y, Z) ;
- #define PROGRESS_MSG(A)
- #define FOPEN(FILENAME, FP, MODE)
- #define FCLOSE(FP) (void) fclose (FP);
- #define SET_COS(W, X, Y, Z)
- #define GET_COS(W, X) (info → cos[W][X].x)
- #define GET_COS_POSITION(W, X) (info → cos[W][X].column)
- #define GET_PROBW1_Z_PREV(X, Y) (info → probw1_z_prev[X ∗ info → m + Y])

- #define GET_PROBW1_Z_CURR(X, Y) (info → probw1_z_curr[X ∗ info → m + Y])
- #define GET_PROBW2_Z_PREV(X, Y) (info → probw2_z_prev[X ∗ info → n + Y])
- #define GET_PROBW2_Z_CURR(X, Y) (info → probw2_z_curr[X ∗ info → n + Y])
- #define GET_PROBZ_PREV(X) (info → probz_prev[X])
- #define GET_PROBZ_CURR(X) (info → probz_curr[X])
- #define GET_PROBZ_W1W2_PREV(W, X, Y) (GET_PROBW1_Z_PREV(W,X) + GET_-PROBW2_Z_PREV(W,Y) + GET_PROBZ_PREV(W))
- #define GET_PROBZ_W1W2_CURR(W, X, Y) (GET_PROBW1_Z_CURR(W,X) + GET_-PROBW2_Z_CURR(W,Y) + GET_PROBZ_CURR(W))
- #define GET_PROB_W1W2(X, Y) (info → prob_w1w2[X ∗ info → n + Y])
- #define logSumsInline(A, B)

Typedefs

- typedef double PROBNODE
- typedef struct cooccur COOCCUR
- typedef struct info INFO

4.15.1 Define Documentation

4.15.1.1 #define BLOCK_HIGH(id, p, n) (BLOCK_LOW ((id)+1, p, n) - 1)

4.15.1.2 #define BLOCK_LOW(id, p, n) ((id) ∗ (n)/(p))

4.15.1.3 #define BLOCK_OWNER(index, p, n) (((p) ∗ ((index)+1)-1)/(n))

4.15.1.4 #define BLOCK_SIZE(id, p, n) (BLOCK_LOW ((id) + 1, p, n)-BLOCK_LOW(id, p, n))

4.15.1.5 #define DBL_LESS(A, B) ((B - A) > DBL_EPSILON)

Test if two double values are close to each other

4.15.1.6 #define DOEXP(X) (expf (X))

Macro to perform the exp function

4.15.1.7 #define DOLOG(X) (logf (X))

Macro to perform a log

4.15.1.8 #define DOLOG1PEXP(x) DOLOGONE(DOEXP(x))

Macro to perform log (1 + expt(x))

4.15.1.9 #define DOLOGONE(X) (log1pf (X))

Macro to perform log (1 + x)

4.15.1.10 #define FCLOSE(FP) (void) fclose (FP);

4.15.1.11 #define FOPEN(FILENAME, FP, MODE)

Value:

```
FP = fopen ((char*) FILENAME, MODE); \
  if (FP == NULL) { \
    fprintf (stderr, "Error %s %s.\n", (strcmp (MODE, "w") == 0) ? "creating" : "opening", FILENAME); \
    exit (EXIT_FAILURE); \
  }
```

4.15.1.12 #define GET_COS(W, X) (info → cos[W][X].x)

Function to retrieve the position from the cooccurrence array

4.15.1.13 #define GET_COS_POSITION(W, X) (info → cos[W][X].column)

Function to retrieve the cooccurrence count from the cooccurrence array

4.15.1.14 #define GET_PROB_W1W2(X, Y) (info → prob_w1w2[X ∗ info → n + Y])

Function to retrieve from P(i,j) – X is w1; Y is w2

4.15.1.15 #define GET_PROBW1_Z_CURR(X, Y) (info → probw1_z_curr[X ∗ info → m + Y])

4.15.1.16 #define GET_PROBW1_Z_PREV(X, Y) (info → probw1_z_prev[X ∗ info → m + Y])

Function to retrieve from P(w1|z); translate 2D to 1D co-ordinates – X is z; Y is w1

4.15.1.17 #define GET_PROBW2_Z_CURR(X, Y) (info → probw2_z_curr[X ∗ info → n + Y])

4.15.1.18 #define GET_PROBW2_Z_PREV(X, Y) (info → probw2_z_prev[X ∗ info → n + Y])

Function to retrieve from P(w2|z); translate 2D to 1D co-ordinates – X is z; Y is w2

4.15.1.19 #define GET_PROBZ_CURR(X) (info → probz_curr[X])

4.15.1.20 #define GET_PROBZ_PREV(X) (info → probz_prev[X])

Function to retrieve from P(z) – X is z

4.15.1.21 #define GET_PROBZ_W1W2_CURR(W, X, Y) (GET_PROBW1_Z_CURR(W,X) + GET_PROBW2_Z_CURR(W,Y) + GET_PROBZ_CURR(W))

4.15.1.22 #define GET_PROBZ_W1W2_PREV(W, X, Y) (GET_PROBW1_Z_PREV(W,X) + GET_PROBW2_Z_PREV(W,Y) + GET_PROBZ_PREV(W))

Function to map P(z|w1,w2) to (P(w1|z) ∗ P(w2|z) ∗ P(z)) – W is z; X is w1; Y is w2

4.15.1.23 #define LN_LIMIT 23.02585093

Accuracy of floating point values as a log (base e) value, multiplied by -1

4.15.1.24 #define logSumsInline(A, B)

Value:

```
{                           \
  register PROBNODE x, y;   \
  if (A > B) {              \
    x = A;  y = B;          \
  }                         \
  else {                    \
    x = B;  y = A;          \
  }                         \
                            \
  /*  a > b  */             \
                            \
  A = (fabs (y - x) > LN_LIMIT) ? x : x + DOLOG1PEXP (y - x);   \
}
```

4.15.1.25 #define MAINPROC 0

ID of the main processor is always 0

4.15.1.26 #define MAX_CLUSTERS 1000

Maximum latent state – value must be a multiple of 10 and the true maximum state is 1 less. Affects the function macro MSG_TAG.

4.15.1.27 #define MIN_PROB (1.0E-24)

Minimum probability

4.15.1.28 #define ML_DELTA 0.001

Minimum difference between two maximum likelihoods

4.15.1.29 #define MPI_TYPE MPI_DOUBLE

Data type to use for probabilities (in MPI functions); must match the definition of PROBNODE

4.15.1.30 #define MSG_RECV_STATUS(V, W, X, Y, Z) ;

4.15.1.31 #define MSG_SEND_STATUS(V, W, X, Y, Z) ;

V is the current process; W is the recipient; X, Y, Z are the iteration, message type, and cluster ID

4.15.1.32 #define MSG_TAG(X, Y, Z) ((X * 10000) + (Y * MAX_CLUSTERS) + Z)

Define the message tag based on the iteration, type of message, and cluster number

4.15.1.33 #define PROGRESS_MSG(A)

Value:

```
if (info -> verbose) { \
  fprintf (stderr, "==\t%s\n", A); \
}
```

Define'd function to indicate program progress

4.15.1.34 #define RANDOM_FLOAT ((PROBNODE)rand () / ((PROBNODE)RAND_MAX + (PROBNODE)1.0))

Generate a random number between [0, 1); cast to floating point first to prevent overflow

4.15.1.35 #define ROUND_DIGITS 100000000

Number of digits to round; used when outputting to binary only

4.15.1.36 #define SET_COS(W, X, Y, Z)

Value:

```
{ \
  info -> cos[W][X].column = Y; \
  info -> cos[W][X].x = Z; \
}
```

Function to retrieve from the cooccurrence array

4.15.1.37 #define TAG_PROBW1_Z 1

4.15.1.38 #define TAG_PROBW1W2 5

4.15.1.39 #define TAG_PROBW2_Z 2

4.15.1.40 #define TAG_PROBZ 3

4.15.1.41 #define TAG_PROBZ_W1W2 4

4.15.2 Typedef Documentation

4.15.2.1 typedef struct cooccur COOCCUR

4.15.2.2 typedef struct info INFO

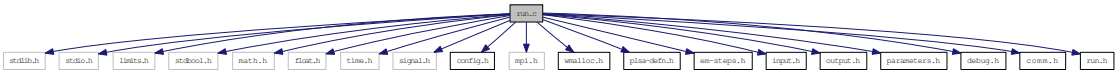
4.15.2.3 typedef double PROBNODE

Data type to use for probabilities

4.16 run.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>
#include <math.h>
#include <float.h>
#include <time.h>
#include <signal.h>
#include "config.h"
#include <mpi.h>
#include "wmalloc.h"
#include "plsa-defn.h"
#include "em-steps.h"
#include "input.h"
#include "output.h"
#include "parameters.h"
#include "debug.h"
#include "comm.h"
#include "run.h"
```

Include dependency graph for run.c:



Functions

- INFO * initialize ()
- void uninitialize (INFO *info)
- bool run (INFO *info)

4.16.1 Function Documentation

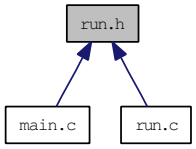
4.16.1.1 INFO* initialize ()

4.16.1.2 bool run (INFO * *info*)

4.16.1.3 void uninitialize (INFO * *info*)

4.17 run.h File Reference

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



Functions

- INFO ∗ initialize ()
- void uninitialize (INFO ∗info)
- bool run (INFO ∗info)

4.17.1 Function Documentation

4.17.1.1 INFO∗ initialize ()

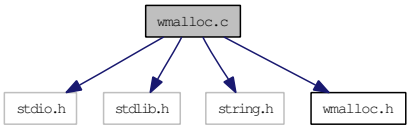
4.17.1.2 bool run (INFO ∗ *info*)

4.17.1.3 void uninitialize (INFO ∗ *info*)

4.18 wmalloc.c File Reference

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

Include dependency graph for wmalloc.c:



Functions

- void * wmalloc (size_t y_arg)
- void * wrealloc (void *x_arg, size_t y_arg)
- void wfree (void *x_arg)
- static unsigned int hash (char *v, signed int M)
- void initWMalloc ()
- void printWMalloc ()
- void printInUseWMalloc (void)
- void countMalloc (void *ptr, size_t amount, const char *file, const unsigned int line)
- void countFree (void *ptr)

Variables

- static unsigned int inuse_malloc = 0
- static unsigned int max_malloc = 0
- static WMSTRUCT ** wm_array
- static char * tempstr

4.18.1 Function Documentation

4.18.1.1 void countFree (void * *ptr*)

4.18.1.2 void countMalloc (void * *ptr*, size_t *amount*, const char * *file*, const unsigned int *line*)

4.18.1.3 static unsigned int hash (char * *v*, signed int *M*) [static]

4.18.1.4 void initWMalloc (void)

4.18.1.5 void printInUseWMalloc (void)

4.18.1.6 void printWMalloc (void)

4.18.1.7 void wfree (void * *x_arg*)

4.18.1.8 void* wmalloc (size_t *y_arg*)

4.18.1.9 void* wrealloc (void * *x_arg*, size_t *y_arg*)

4.18.2 Variable Documentation

4.18.2.1 unsigned int inuse_malloc = 0 [static]

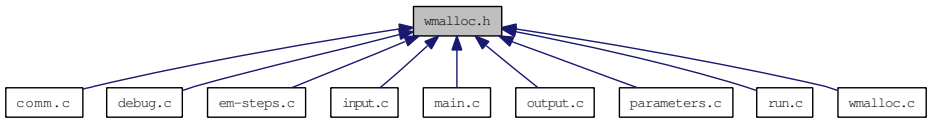
4.18.2.2 unsigned int max_malloc = 0 [static]

4.18.2.3 char* tempstr [static]

4.18.2.4 WMSTRUCT** wm_array [static]

4.19 wmalloc.h File Reference

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



Data Structures

- struct wmstruct

Defines

- #define WM_SIZE 65536
- #define TEMPSTRLEN 80

Typedefs

- typedef struct wmstruct WMSTRUCT

Functions

- void * wmalloc (size_t y_arg)
- void * wrealloc (void *x_arg, size_t y_arg)
- void wfree (void *x_arg)
- void initWMalloc (void)
- void printWMalloc (void)
- void printInUseWMalloc (void)
- void countMalloc (void *ptr, size_t amount, const char *file, unsigned int line)
- void countFree (void *ptr)

4.19.1 Define Documentation

4.19.1.1 #define TEMPSTRLEN 80

4.19.1.2 #define WM_SIZE 65536

4.19.2 Typedef Documentation

4.19.2.1 typedef struct wmstruct WMSTRUCT

4.19.3 Function Documentation

4.19.3.1 void countFree (void * *ptr*)

4.19.3.2 void countMalloc (void * *ptr*, size_t *amount*, const char * *file*, unsigned int *line*)

4.19.3.3 void initWMalloc (void)

4.19.3.4 void printInUseWMalloc (void)

4.19.3.5 void printWMalloc (void)

4.19.3.6 void wfree (void * *x_arg*)

4.19.3.7 void* wmalloc (size_t *y_arg*)

4.19.3.8 void* wrealloc (void * *x_arg*, size_t *y_arg*)

Index