付録

モジュールの階層

version7 のプログラムはモジュール別になっており，次のような構造に分けられています．

アプリケーション	実行プログラムです．
意味モデルライブラリ	意味モデル用のライブラリです．
拡張 C ライブラリ	意味モデルに限らない C のライブラリです．

階層の上位のものが下位のものを呼び出すことはありますが，その逆はありません．

アプリケーション

意味モデルライブラリを使用した実行プログラムです．

主に専用の Makefile によって順番に実行されていきます．

次のものがあります．

featurec	特徴語を決めます．
filterc	辞書読み取りのフィルターのキャッシュを作成します．
filters	辞書テキストをフィルターに通し，A 行列を作成します．
compo	複数の単語の定義を合成し，一つの単語にします．
normal	正規化を行います．
space	A 行列から意味空間を作成します．
proj	射影を行います．
rsort	高速検索アルゴリズムの為のソート行列を作成します．
lmmc	意味的距離計算のサンプルプログラムです．

意味モデルライブラリ

意味モデル用の関数群です．目的別に次のようなものに分けられています．

libmmm.a	意味モデルアプリケーション用の一般関数群です．
libmmath.a	意味モデルアプリケーション用の数値計算用関数群です．
libcache.a	意味モデルアプリケーション用のキャッシュ関数群です．
libcalc.a	意味モデル応用アプリケーション用の関数群です．

拡張 C ライブラリ

意味モデルに限らない一般用の関数群です．目的別に次のように分けられています．

libcE.a	標準の関数にエラーチェックを追加した関数群です．
libcext.a	標準の関数を特定の使い方の為に使いやすくした関数群です．
libstra.a	文字列のリスト構造を扱う関数群です．
libbfs.a	ファイル名を base_dir,filename,suffix の 3 つに分けて扱う為の関
libapp.a	雑多な関数群です．

距離計算ライブラリ

version7 では多種多様なアプリケーションが意味モデルを容易に扱えるように距離計算の部分をライブラリ形式にしています．意味モデルを扱うアプリケーションは，このライブラリをリンクする事で意味モデルの距離計算を行う事が出来ます．

距離計算ライブラリで扱う型

距離計算ライブラリで使用する変数の型には，大きく分けて次のものがあります．

MMM_ENV	動作環境を指定します．
MMM_SPACE	空間情報を扱います．
MMM_MATRIX	座標データを扱います．

MMM_KC	キーワードと文脈を扱います.
MMM_RESULTS	検索結果を扱います.

MMM_ENV

struct mmm_env は距離計算ライブラリの実行環境を設定する変数の型です. この構造体は以下の構造体から成っています.

MMM_FILENAMES	距離計算を行うデータの細かいファイル名を設定します.
MMM_VALUES	距離計算を行う時の各種パラメータを設定します.
MMM_FUNCTIONS	距離計算を行う時に呼び出される関数を設定します.

デフォルト値は init_env() で設定されます.

MMM_FILENAMES

struct mmm_filenames は距離計算ライブラリが扱うファイルのファイル名を指定します. 内容は以下の通りです. ユーザは通常, これらを変更する必要はありません.

in_dir	行列データディレクトリ名
space_dir	空間データディレクトリ名
features	特徴語の名前のファイル名
feature_size	特徴語の数のファイル名
feature_ID	特徴語のＩＤのファイル名
feature_ptr	特徴語のポインタのファイル名
words	見出し語のファイル名
word_size	見出し語の数のファイル名
word_ID	見出し語のＩＤのファイル名
word_ptr	見出し語のポインタのファイル名

a	A行列のファイル名
a_ID	A行列のＩＤのファイル名
q	射影行列のファイル名
q_size	空間の広さのファイル名
q_ID	射影行列のＩＤのファイル名
p	射影後の行列のファイル名
p_ID	射影後の行列のＩＤのファイル名
r	ソートの逆参照ポインタのファイル名
r_ID	逆参照ポインタのＩＤのファイル名
in_cache_dir	データキャッシュのディレクトリ名
space_cache_dir	空間データキャッシュのディレクトリ名
txt	テキストファイルの拡張子
xdr	ＸＤＲファイルの拡張子
cache	キャッシュファイルの拡張子
t_words	データ行列のタグ名
t_space	空間データ行列のタグ名

MMM_VALUES

struct mmm_values は距離検索時に与えるパラメータを指定します．内容には以下のものがあります．

es	ε_s

MMM_FUNCTIONS

距離計算ライブラリで呼び出される関数を指定します．ユーザは，距離計算ライブラリで用意されている他の関数や自分で定義した関数を再指定する事で，距離計算の方法を変更する事が出来ます．変更が可能な関数は以下の通りです．

kproj	キーワードを射影する関数
calc	距離計算を行い，指定個数の解を返す関数
dist	キーワードと指定単語の距離を返す関数
dcmp	結果をソートする時の比較関数

引数などの指定方法はデフォルトで指定されている関数を参照して下さい．

MMM_SPACE

struct mmm_space は距離計算を行う空間を扱う型です．init_space() で初期化を行い，load_space() で指定ディレクトリから空間データを読み込みます．読み込んだ空間情報の一部は load_matrix() で行列データに複写され，kc_malloc_a() でキーワードデータにも複写されます．内容は以下の通りです．

qsize	次元数
fsize	特徴語数
q	射影行列
f	特徴語
fmx	最大特徴語名長
fid	feature_ID
qid	q_ID
path	data directory path

MMM_MATRIX

struct mmm_matrix は座標データが入った行列を扱う型です．具体的には，キーワードやコンテキストに使われる単語群や，検索の対象となる単語群です．init_matrix() で初期化を行い，load_matrix() で指定ディレクトリから座標データを読み込みます．内容は以下の通りです．

wsize	定義語数

qsize	次元数
w	見出し語名
p	射影後のベクトル
r	高速検索用逆参照ポインタ
fg	有効単語フラグ
fsize	特徴語数
a	A 行列
f	特徴語
wmx	最大単語名長
fmx	最大特徴語名長
wid	word_ID
pid	p_ID
rid	r_ID
aid	a_ID
fid	feature_ID
path	data directory path

MMM_KC

struct mmm_kc は距離計算に使われるキーワードとコンテキストを扱う型です．init_kc() で初期化を行い，kc_malloc_a() で領域を確保してから使用します．内容は以下の通りです．

fsize	特徴語の数
ka	キーワードのA行列
ca	文脈のA行列
qsize	空間の広さ
kp	キーワード射影後のベクトル
cp	文脈語の射影後のベクトル
q	射影行列
sq	部分空間

MMM_RESULTS

struct mmm_results は距離計算の結果を扱う型です．init_results() で初期化し，results_malloc() で領域を確保して使用します．内容は以下の通りです．

qnsize	返して欲しい部分空間情報の次元数
qn	返ってくる部分空間情報
resize	返して欲しい結果の個数
res	結果

MMM_SK

struct mmm_sk は距離計算ライブラリ内部で使用される作業用の変数を扱う型です．普通に使用してる限り扱う必要はありませんが，自作の距離計算関数を定義する時などには扱わねばなりません．内容は以下の通りです．

fsize	特徴語数
qsize	空間の広さ
ca	文脈語のA行列
cp	射影後の文脈語
ka	キーワードのA行列
kp	射影後のキーワード
q	射影行列
qs	部分空間に選ばれたかのフラグ
qn	部分空間のソート済み軸番号
gsize	部分空間の大きさ
gs	部分空間の重みと軸番号
wsize	比較対象語の行列の全見出し語数
resize	結果の個数

res　　結果

距離計算ライブラリを使用したプログラムの流れ

距離計算ライブラリを使用したプログラムは以下の様な流れになることになります.

距離計算ライブラリ変数の初期化

init_env(),init_space(),init_matrix(),init_kc(),init_results() などを呼びだし, 使用する構造体を初期化します.

空間情報の読み込み

load_space() を呼びだし, 空間情報をファイルから読み込みます.

座標情報の読み込み

load_matrix() を呼びだし, キーワード, 文脈に使用される単語の座標情報と検索対象語の座標情報を読み込みます.

検索対象語の有効フラグの設定

fg_malloc() を呼びだして, 検索対象語の有効フラグ領域を確保し, aset_mat_fg() を呼び出して全ての検索対象語を有効にします. aclr_mat_fg() を呼び出して全検索対象語を無効にし, add_target_matw() で指定する事もできます.

キーワードと文脈語のワークを確保

kc_malloc() を呼び出してキーワードと文脈を指定する為の領域を確保します.

検索結果用の領域を確保

results_malloc() を呼び出して検索結果の為の領域を確保します. この時, 検索結果の個数を指定します.

各種パラメータの設定

aset_kc_sq(),pclr_kc_sq() を呼びだし，主軸抜きの設定をします．また，コールバック関数を変更することにより，キーワード分解や検索方法，距離定義などを変更する事ができます．同様に ε_s などの設定も行います．

キーワードと文脈の設定

clear_keyword() でキーワード定義をクリアしてから set_keyword_matw() でキーワードを設定します．また，clear_context() で文脈定義をクリアしてから add_context_matw(で文脈語群を追加していきます．

検索を行う

semap() 関数を呼びだして検索を行います．

距離計算ライブラリの関数マニュアル

距離計算ライブラリには以下の関数が定義されています．

変数の初期化を行う関数

距離計算ライブラリで引数として渡される変数の初期化を行う関数群です．関数の型に合わせてそれぞれ用意されています．距離計算ライブラリで使用する変数は必ず初期化して下さい．

init_env

```
init_env(env)
  MMM_ENV *env;
```

*env で与えられた距離計算ライブラリの環境を設定します．この時に各種パラメータは以下のように初期化されます．これ以外の設定で距離計算を行いたい時は，パラメータを変更関数を使用してパラメータを変更してから使用して下さい．

v.es	0.0 ($\varepsilon_s = 0.0$)

c.kproj	kprojr (キーワード分解あり)
c.calc	stdcalc (単純距離計算アルゴリズム)
c.dist	stddist (重み付き 2 ノルム)
c.dcmp	srvcmp (距離の近い順)

init_space

```
init_space(space)
  MMM_SPACE *space;
```

*space で与えられた空間情報を初期化します. 誤動作を防止するだけで特に意味のある値は設定しません.

init_matrix

```
init_matrix(mat)
  MMM_MATRIX *mat;
```

*mat で与えられた行列情報を初期化します. 誤動作を防止するだけで特に意味のある値は設定しません.

init_kc

```
init_kc(kc)
  MMM_KC *kc;
```

*kc で与えられたキーワードと文脈情報を初期化します. 誤動作を防止するだけで特に意味のある値は設定しません.

init_results

```
init_results(rs)
  MMM_RESULTS *rs;
```

*rs で与えられた検索結果格納情報を初期化します. 誤動作を防止するだけで特に意味のある値は設定しません.

ファイルを読み込む関数

生成した空間，定義した比較対象語，定義したキーワードや文脈を読み込む関数群です．

load_space

```
load_space(env,space,dir,cdir)
  MMM_ENV *env;
  MMM_SPACE *space;
  char *dir;
  char *cdir;
```

*env で与えられた環境の下で(ファイル名など)，*dir で指定されたディレクトリから空間情報を *space に読み込みます．その時，キャッシュディレクトリとして，*cdir で指定されたディレクトリを使用します．通常 *dir と *cdir は同じディレクトリを指定します．

load_matrix

```
load_matrix(env,space,mat,dir,cdir)
  MMM_ENV *env;
  MMM_SPACE *space;
  MMM_MATRIX *mat;
  char *dir;
  char *cdir;
```

*env で与えられた環境の下で(ファイル名など)，*space の空間上に配置されている，*dir に存在する行列情報を *mat に読み込みます．その時，キャッシュディレクトリとして，*cdir で指定されたディレクトリを使用します．通常 *dir と *cdir は同じディレクトリを指定します．なお，load_matrix 関数は空間情報を必要とするので，事前に load_space 関数を呼び出している必要があります．この関数を呼び出すと，空間情報の一部が *mat に複写されます．

ワークを確保する関数

fg_malloc

```
fg_malloc(mat)
  MMM_MATRIX *mat;
```

*mat で与えられた行列情報の有効フラグ用の領域を確保します．この領域を確保しないと有効フラグを使用する事はできません．

kc_malloc

```
kc_malloc_a(kc,space)
  MMM_KC *kc;
  MMM_SPACE *space;
```

*kc で与えられたキーワード・文脈用の変数領域に *space で与えられた空間情報を持ちいて，キーワード・文脈用のベクトルの領域を確保します．また，軸有効フラグ用の領域も確保されます．この関数を呼び出すことにより，空間情報の一部が複写されます．

results_malloc

```
results_malloc(rs,qnn,rsn)
  MMM_RESULTS *rs;
  int qnn;
  int rsn;
```

*rs で与えられた検索結果用の変数領域に，qnn 本の軸情報と rsn 個の検索結果を確保できる領域を作成します．検索時には，この関数で指定した個数よりも多い情報を受け渡すように指示してはいけません．

フラグを扱う関数

比較対象語や軸には，１つづつフラグが附属し，有効か無効かを指定する事ができます．それらを扱う関数群です．

aset_mat_fg

```
aset_mat_fg(mat)
  MMM_MATRIX *mat;
```

*mat で与えられた行列の全ての有効フラグをセットします．通常，比較対象語の行列に対して使用し，全ての単語が有効であることを指定します．事前に fg_malloc 関数を呼び出して領域を確保しておく必要があります．

aclr_mat_fg

```
aclr_mat_fg(mat)
  MMM_MATRIX *mat;
```

*mat で与えられた行列の全ての有効フラグをクリアします．通常，比較対象語の行列に対して使用します．この関数を呼び出して全ての単語を一旦無効にしてから add_target_matw 関数で有効な単語を１つづつ指定してゆくという使われ方をします．事前に fg_malloc 関数を呼び出して領域を確保しておく必要があります．

add_target_matw

```
int add_target_matw(mat,tw)
  MMM_MATRIX *mat;
  char *tw;
```

*mat で与えられた行列の有効フラグに *tw で指定された比較対象語を追加します．成功すれば０が，指定された単語が存在しない理由で失敗すれば −1 が返されます．

aset_kc_sq

```
aset_kc_sq(kc)
  MMM_KC *kc;
```

*kc で与えられたキーワード・文脈用の情報領域に存在する，全ての軸有効フラグをセットします．事前に kc_malloc 関数を呼び出して領域を確保しておく必要があります．

pclr_kc_sq

```
pclr_kc_sq(kc,n)
  MMM_KC *kc;
  int n;
```

*kc で与えられたキーワード・文脈用の情報領域に存在する，n で指定された軸有効フラグをクリアします．n に 0 を指定することにより，主軸抜きを行う事ができます．事前に kc_malloc 関数を呼び出して領域を確保しておく必要があります．

pset_kc_sq

```
pset_kc_sq(kc,n)
  MMM_KC *kc;
  int n;
```

*kc で与えられたキーワード・文脈用の情報領域に存在する，n で指定された軸有効フラグをセットします．通常，使用することはありません．事前に kc_malloc 関数を呼び出して領域を確保しておく必要があります．

キーワード・文脈設定関数

clear_keyword

```
clear_keyword(kc)
  MMM_KC *kc;
```

*kc で与えられたキーワード・文脈用の変数のキーワード情報を零ベクトルに初期化します．

clear_context

```
clear_context(kc)
  MMM_KC *kc;
```

*kc で与えられたキーワード・文脈用の変数の文脈情報を零ベクトルに初期化します．

set_keyword_matw

```
int set_keyword_matw(kc,mat,kw)
  MMM_KC *kc;
  MMM_MATRIX *mat;
  char *kw;
```

*kc で与えられたキーワード・文脈用の変数のキーワードベクトルに *mat で与えられた行列情報の中の単語 *kw のベクトルを設定します．成功すれば 0 が，指定された単語が存在しない理由で失敗すれば −1 が返されます．

add_context_matw

```
int add_context_matw(kc,mat,cw)
  MMM_KC *kc;
  MMM_MATRIX *mat;
  char *cw;
```

*kc で与えられたキーワード・文脈用の変数の文脈ベクトルに *mat で与えられた行列情報の中の単語 *cw のベクトルを追加します．成功すれば 0 が，指定された単語が存在しない理由で失敗すれば −1 が返されます．

検索結果指定関数

set_rs_nq

```
set_rs_nq(rs,nq)
  MMM_RESULTS *rs;
  int nq;
```

*rs で与えられた検索結果領域に書き込む重みの強い軸の情報の個数を nq 個に指定します．nq は確保された領域の個数以下でなければなりません．

set_rs_nr

```
set_rs_nr(rs,nr)
  MMM_RESULTS *rs;
```

```
int nr;
```

*rs で与えられた検索結果領域に書き込む検索結果の個数を nr 個に指定します．nr は確保された領域の個数以下でなければなりません．

パラメータ設定関数

env_kreproj_on

```
env_kreproj_on(env)
  MMM_ENV *env;
```

*env で与えられた環境の「キーワード分解あり」のフラグをセットします．

env_kreproj_off

```
env_kreproj_off(env)
  MMM_ENV *env;
```

*env で与えられた環境の「キーワード分解なし」のフラグをセットします．

set_env_es

```
set_env_es(env,es)
  MMM_ENV *env;
  double es;
```

*env で与えられた環境の ε_s を es に設定します．

set_env_calc

```
set_env_calc(env,calcf)
  MMM_ENV *env;
  int (*calcf)();
```

*env で与えられた環境で，部分空間を計算した後，計算を行う関数を (*calcf)() に設定します．引数に与えることのできる関数として，ライブラリには以下の関数が用意されています．

stdcalc	全ての比較対象語との距離を計算し，ソートを行い解を求める．
hscalc1	キーワード近くの比較対象語から近い順に距離を計算し，解を求める．
hscalc2	キーワード近くの比較対象語から近い順に距離を計算し，解を求める．

set_env_dist

```
set_env_dist(env,distf)
  MMM_ENV *env;
  int (*distf)();
```

*env で与えられた環境で，キーワードと比較対象語の距離計算を行う関数を (*distf)() に設定します．引数に与えることのできる関数として，ライブラリには以下の関数が用意されています．

stddist	文脈による重みつき空間内で，キーワードと比較対象語との距離を求
normdist1	文脈ベクトルと同じ方向の成分だけからなる原点からの距離を求めま

set_env_dcmp

```
set_env_dcmp(env,dcmpf)
  MMM_ENV *env;
  int (*dcmpf)();
```

*env で与えられた環境で，検索結果を求めた後，ソート時の比較関数を (*dcmpf)() に設定します．引数に与えることのできる関数として，(version7-lib) ライブラリには以下の関数が用意されています．

srvcmp	距離の小さい順にソートと行います．
srvrcmp	距離の大きい順にソートと行います．

検索実行関数

semap

```
semap(env,rs,kc,mat)
  MMM_ENV *env;
  MMM_RESULTS *rs;
  MMM_KC *kc;
  MMM_MATRIX *mat;
```

*env で与えられた環境の上で *rs で与えられた検索結果変数領域に *kc で与えられたキーワードと文脈を使用して *mat で与えられた行列を対象に連想検索を行います.

検索結果を取り出す関数

rs_qn_n

```
int rs_qn_n(rs,n)
  MMM_RESULTS *rs;
  int n;
```

*rs で指定された検索結果情報からn 番目に重みの強い軸の番号を返します.

rs_qn_d

```
double rs_qn_d(rs,n)
  MMM_RESULTS *rs;
  int n;
```

*rs で指定された検索結果情報からn 番目に重みの強い軸の重みを返します.

rs_res_n

```
int rs_res_n(rs,n)
  MMM_RESULTS *rs;
  int n;
```

*rs で指定された検索結果情報からn 番目の検索結果の番号を返します. 返された番号は mat_word 関数を通す事により単語に変換する事ができます.

rs_res_dt

```
double rs_res_dt(rs,n)
  MMM_RESULTS *rs;
  int n;
```

*rs で指定された検索結果情報からｎ番目の検索結果の距離を返します.

rs_res_cost

```
int rs_res_cost(rs,n)
  MMM_RESULTS *rs;
  int n;
```

*rs で指定された検索結果情報からｎ番目の検索結果のコストを返します.

行列情報を取り出す関数

space_qsize

```
int space_qsize(space)
  MMM_SPACE *space;
```

*space で与えられた空間の次元数を返します.

mat_wsize

```
int mat_wsize(mat)
  MMM_MATRIX *mat;
```

*mat で与えられた行列の見出し語数を返します.

mat_word

```
char *mat_word(mat,n)
  MMM_MATRIX *mat;
```

*mat で与えられた行列のｎ番目の見出し語を返します.

論文リスト

宮原 隆行, 清木 康, 北川 高嗣,
「意味の数学モデルによる意味的連想検索の高速化アルゴリズムとその実現方式」,
情報処理学会論文誌,
Vol.38, No.7, pp.1399-1411 , July. 1997.

Y. Kiyoki, T. Kitagawa and T. Miyahara, "A fast algorithm
of semantic associative search for databases and knowledge bases,"
Information Modelling and Knowledge Bases (IOS Press), Vol. VII,
pp. 44-58, 1996.

宮川明子, 清木康, 宮原隆行, 北川高嗣,
「画像データベースを対象とした意味的連想検索の高速化アルゴリズム」,
情報処理学会論文誌, to appear.