Outline

1. Introduction
   - What is MetaCoDe
   - Main Advantages & Drawbacks
   - Content of the Package

2. Requirements

3. Installation
   - General Utilities
   - UMLS Resources
   - The package

4. Working Example

5. Conclusion
Outline

1. Introduction
   - What is MetaCoDe
   - Main Advantages & Drawbacks
   - Content of the Package

2. Requirements

3. Installation
   - General Utilities
   - UMLS Resources
   - The package

4. Working Example

5. Conclusion
What is MetaCoDe

- A suite of program utilities
- That extract UMLS tags from biomedical texts
- That are written in Perl and C++
- That have been used only on Linux, but should run everywhere as it does not rely on any platform specific features
- That are byproducts of Question-Answering related project
Outline

1. Introduction
   - What is MetaCoDe
   - Main Advantages & Drawbacks
   - Content of the Package

2. Requirements

3. Installation
   - General Utilities
   - UMLS Resources
   - The package

4. Working Example

5. Conclusion
Main Advantages & Drawbacks

Well, for sure there are a lot of shortcomings that should be fixed. Up to now, this tool has not been really used, and so has been lacking user feedback and improvement opportunities, at least so far. As a matter of fact, it is a by product of another project; this is a reason for some of its peculiarities.
Main Advantages & Drawbacks

Advantages

- GPL license
- Simple logic and short programs, so easy to modify
- A browser to inspect the results
- Should run quickly enough
- Multilingual: able to process corpus in French or in English, and should easily be extended to other languages (the corpus are to be monolingual, nevertheless)
Main Advantages & Drawbacks

- No user feedback up to now
- designed to tag sets of short texts, for example bunches of abstracts or patient records
- still requires some handwork from the user (no GUI, nor unique script)
- ...
Outline

1. Introduction
   - What is MetaCoDe
   - Main Advantages & Drawbacks
   - Content of the Package

2. Requirements

3. Installation
   - General Utilities
   - UMLS Resources
   - The package

4. Working Example

5. Conclusion
Content of the Package

The package is made of 4 directories:

- doc : where the documentation & license files are;
- progs : where the source code & Perl scripts are
- examples : some data used along this guide
Content of the Package

Doc

- *guide.pdf*: what you are reading now
- *LICENSE*: the GPL license
Content of the Package

progs

- *explode.pl*: preprocessing script
- *concat.pl*: preprocessing script
- *extract-UMLS-resources.pl*: preprocessing script
- *tag.cpp*: the tagger
- *makefile*: an ultrabasic makefile (one line!) just to compile the above file
- *populate-metacode.pl*: postprocessing script
- *populate-resumes.pl*: postprocessing script
- *browse-bilingue.pl*: a browser to inspect the result (don’t mind the strange name ;-) )
**examples**

- `example.sh`: script running the demo. You may modify it if some of the assumptions made for running the example are not verified.
- `config.d`: contains configuration files
- `data`: the corpus & output files
  - `resumes.en`: corpus (36 abstracts)
- `work`: for outputs
Requirements

- C++ compiler & linker, GNU g++ for example
- pcre library and header files, available at http://sourceforge.net/projects/pcre/files/ (version 8.02 is fine)
- a relational data base; mysql is advised though the use of any other RDBMS should be possible upon slight modification. Throughout this guide we will assume the use of mysql.
- a Perl interpreter, and some perl modules available from CPAN: Getopt::Long, Term::ReadKey, Tk, Tk::Adjuster, DBI, DBD::mysql (or whatever).
- UMLS data bases; you must get an UMLS License to be able to use this resource. Please refer to http://umlsks.nlm.nih.gov if you need it.
- bash interpreter along some common Unix tools (such as find, sort or grep) will be useful as well.
The following part of the document describes the installation steps you will have to go through in order to be able to run MetaCoDe. Of course this is to be done once only. Please pay a special attention to the UMLS part, and more precisely to the creation of the tables, as the system will heavily rely on these objects.
We do not use this OS, but if you are running Windows, you may consider using cygwin (www.cygwin.com) for the development tools and Unix typical utilities, unless of course you already have all you need on your machine. ActiveState’s Perl (that you can get from a community edition) is also an option (see http://www.activestate.com.) Furthermore mysql comes with an MSI installer. Otherwise, all the utilities (mysql, g++, Perl) are available in every standard Linux distribution, and so it should be for BSD UNIXes. So if you are living in the best of the worlds, you should have everything you need under your fingers. If something is missing, well, go to install it, this is not more difficult than under Windows.
Just one point about mysql. You will have to use an account identified by a password, with grants allowing to create databases and to alter tables.
tree-tagger is a Part of Speech tagger. You can download it from http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/. All you need to do is untar the archive that contains the executable file and to update your PATH environment variable. You will need also the English parameter file available at the same location.
The first step is to get the UMLS data bases from http://umlsks.nlm.nih.gov and to install them with MetamorphoSys. Please refer to UMLSKS documentation for that. During installation, you are free to choose the sources you want. If you exclude some sources (for example SNOMED), the only impact will be that MetaCoDe will not extract the corresponding tags from the corpora. There is no need to install the MRCOOC table, and you can keep only English sources.
The more important point here is to install the tables in mysql (MetamorphoSys is able to produce the required SQL scripts for that, if you ask for them). As a matter of fact, running MetaCoDe will only require the following tables: MRSTY, MRCONSO, MRDEF, MRXW_EN (or MRXW_XYZ if you want to tag a corpus written in XYZ.) So you can spare some disk space if necessary.
Once the UMLS tables are populated, three indexes must be created in order to speed up the corpora processing. These indexes are the followings:

- on SUI for MRWX_EN and MRCONSO;
- on WD for MRWX_EN;

Indexes creation

1. `CREATE INDEX MRWX_EN_SUI_IDX ON MRWX_EN(SUI);`
2. `CREATE INDEX MRCONSO_SUI_IDX ON MRCONSO(SUI);`
3. `CREATE INDEX MRWX_EN_WD_IDX ON MRWX_EN(WD);`
It is possible that creating an index on WD field of MRWX_ENG doesn't work at first since it has been created with the text type. If so, you can change the type of this attribute before indexing:

**WD type setting**

```sql
1 select max(LENGTH(wd)) from MRWX_ENG;
% Assuming this gives 80:
3 alter table MRWX_ENG CHANGE WD WD VARCHAR(80);
```
The final step in UMLS resources installation is to create an extra table by hand. This table must keep the length of each concept string.

### String Length Table

```
create table SUILEN_ENG as select SUI, count(*) as LEN
from MRXW_ENG group by SUI;
```

Note that this should be done for any other corpora languages (eg. French) if MetaCoDe is to be applied to these corpora as well.
Outline

1. Introduction
   - What is MetaCoDe
   - Main Advantages & Drawbacks
   - Content of the Package

2. Requirements

3. Installation
   - General Utilities
   - UMLS Resources
   - The package

4. Working Example

5. Conclusion
It comes as a tarball. All you need is to untar it in any location of your choice, and then to compile tag.cpp with make for example.
In the remaining part of this guide, we will illustrate the use of MetaCoDe through an example. All the necessary data needed to run the example come with the package so you will be able to try it by yourself.

We will assume the following facts:

- that you have installed all required elements, and that you are using mysql;
- that you have installed the UMLS tables in the database named UMLS_2009_AA;
- that the mysql account name is 'thierry';
- that you are using english-par-linux-3.1.bin english tree-tagger parameter file, and that it has been copied in /usr/share/treetagger;
- that you are using bash as your shell interpreter.
A MetaCoDe run consists of the following steps:

- create a project directory under which there should be a *resumes.en* (this name is mandatory!) sub-directory containing the corpora, one file per abstract (or corpora unit); in our case, the project directory is *examples*;
- apply the part-of-speech tagger to the corpus;
- extract the necessary UMLS resources;
- tag the corpora;
- eventually browse it.

The *examples* directory contains *example.sh*, which is able to run the whole demo. You are advised to read this file, which may contain extra useful information.
To run this example by yourself you must go to the examples directory. You will have to do some slight modifications in the parameter files that are in the config.d directory:

- change `dbi-user-name` property in `config.d/extract-used-wd.param`;
- change `dbi-dns` property in `config.d/extract-used-wd.param` if you are not using mysql, or if you did not install UMLS tables in UMLS_2009_AA;
- specific changes will be needed in `config.d/config.en`, for processing your own corpora. See the appendix part of the guide.
In the following we will use the following variables:

- **SOURCE_DIR**: location of the original corpus;
- **PROG_DIR**: location of the scripts;
- **DEST_DIR**: location of intermediate files;
- **CONFIG_DIR**: location of configuration files;
- **TREETAG_CMD**: tree-tagger command;

Please have a look in *example.sh* to see how they are initialized.
Part-of-speech tagging

```bash
find $SOURCE_DIR -type f | perl $PROG_DIR/explode.pl -fl -di . -do $DEST_DIR -vf -pnt NP
find $DEST_DIR -name ".e" | perl $PROG_DIR/concat.pl > $DEST_DIR/pn.concat
$TREETAG_CMD < $DEST_DIR/pn.concat > $DEST_DIR/pn.concat.tag
```

If this step succeeds, you should get a `pn.concat.tag` file in your work directory. This is the result of tree-tagging the corpus. Each unit (let say abstract) specific part is identified with `<__ORIGIN__ path="...">` delimiters that identify the source of the data.
Resource Extraction

perl $PROG_DIR/extract-UMLS-resources.pl -param $CONFIG_DIR/extract-used-wd.param -filein $DEST_DIR/pn.concat.tag -prefix $DEST_DIR/en.

Beware that here the program will ask you for a password (more info in example.sh). This step will require some slight modification in extract-used-wd.param, but they are straightforward.

If successful, you will get some extracted resource files in your work directory: en.CUISTY en.SUICUI en.SUILENGTH and en.WDSUI.
Note that the overhead of the Resource Extraction step is quite big compared to the size of our little example data. It is more acceptable when really big corpora are processed. Nevertheless, this step is nowadays questionable. It was useful when memory amount of common machines was limited. But today you can find cheap laptops with 8Go main memory in every hardware store near your door. I guess that in a near future this step will just disappear...
$PROG_DIR/tag $CONFIG_DIR/config.en > $DEST_DIR/result

This is the last step of the tagging process. The program uses the files built at the previous steps (referenced in config.d/config.en configuration file) to produce the resulting tagging, which is here work/result.
An output is made of three or four fields:

```
Extract of the result file
0 1 <_ORIGIN_ path="work/data/resumes.en/9459469.e">
1 1 [CONTEXT] 1:S1430531.C0542559(T169)
2 1 :
3 1 The
4 7 [spread][of][drug][-][resistant][Streptococcus][pneumoniae] 116:S5879296.C1444105(T007)
11 1 in
12 1 the
13 1 [community] 1:S0026824.C0009462(T083)
...
```

The first field is the position of the fragment in the text (in token number); the second field is the size of the fragment; the third one is the fragment itself. When the fragment has been identified as a nominal phrase, its compounds are enclosed in square brackets []. Eventually the UMLS tags are in the forth field. Note that if the first field is equal to 0, then the line contains information about the origin of the following data.
Consider the following extract from the result:

```
4 7 [spread][of][drug][-][resistant][Streptococcus][pneumoniae] 116:S5879296.C1444105(T007)
```

Here the tag is 116:S5879296.C1444105(T007), meaning that:

- the UMLS concept (CUI) assigned to a part of this phrase is C1444105;
- the String Id (SUI) is S5879296;
- this correspond to the semantic type T007;
- \(116 = 64 + 32 + 16 + 4 = 1110100_2\); this is a mask allowing to know exactly which part of the fragment is targeted by the concept. Here the mask identifies [drug], [resistant], [Streptococcus] and [pneumoniae] (the mask is to be read in inverse order). As a matter of fact, C1444105 is “Drug resistant Streptococcus pneumoniae” according to the Metathesaurus.

It is possible, and it happens very often, that more than one tag are assigned to a fragment. The mask allows you to know to which part, may be not contiguous, a given tag was assigned.
The browser of MetaCoDe requires the result of the tagging to be installed in a database. For that, we provide two scripts that help to populate tables with the original texts on the one hand, and with the results of the tagging on the other hand.

### Preparation before browsing

```
perl $PROG_DIR/populate-metacode.pl -lang EN -filein $DEST_DIR/result
   -tags $DEST_DIR/tags.txt -values $DEST_DIR/val.txt -sql $DEST_DIR/poptag.sql
ls $SOURCE_DIR > $DEST_DIR/keys.txt
perl $PROG_DIR/populate-resumes.pl -key-file $DEST_DIR/keys.txt -row-key $DEST_DIR/rowkeys.txt
   -en-data $DEST_DIR/data.txt -sql $DEST_DIR/poptxt.sql
mysql --password

mysql> create database example;
mysql> use example
mysql> source work/poptag.sql
mysql> source work/poptxt.sql
mysql> quit
```
Browsing a Tagged Corpus

Browsing

To launch the browser, just enter the following command line (assuming the names of the data bases):

```
   -user [the user name] -pwd [the password] -nommtx -nomtcdfr &
```

The -nommtx and -nomtcdfr flags are necessary to switch off some specific features of the browser (remember that it comes from a wider project). Without these flags, the browser will not load, as it will not find some extra expected information.
The browser links the tagged corpus with the Metathesaurus, making it easier to evaluate the result of a tagging. It displays extracted concepts, and interactively gets the concepts definitions and synonymous from the Metathesaurus. It allows you to navigate between text units (abstracts, for example), thanks to the buttons at the bottom of the frame window. It uses four panels:

- **the upper panel displays the current text unit.** Colored parts shows the extracted nominal phrases (red or blue are there just to help distinguish between nearby phrases). Clicking on a colored part make the assigned tags to be displayed in the lower panels.

- **the middle panel displays all the UMLS concepts that were identified in the current text unit.** Clicking on a line of this panel makes the definition of the corresponding UMLS concept to appear the the lower right panel.

The browser links the tagged corpus with the Metathesaurus, making it easier to evaluate the result of a tagging. It displays extracted concepts, and interactively gets the concepts definitions and synonymous from the Metathesaurus. It allows you to navigate between text units (abstracts, for example), thanks to the buttons at the bottom of the frame window. It uses four panels:

- **the upper panel displays the current text unit.** Colored parts shows the extracted nominal phrases (red or blue are there just to help distinguish between nearby phrases). Clicking on a colored part make the assigned tags to be displayed in the lower panels.

- **the middle panel displays all the UMLS concepts that were identified in the current text unit.** Clicking on a line of this panel makes the definition of the corresponding UMLS concept to appear the the lower right panel.

Thierry Delbecque

MetaCoDe V0.1 User Guide
the bottom left panel displays the UMLS concepts that were extracted from the phrase the user has selected in the upper panel. It shows the phrase, the extracted CUI’s, for each extracted CUI its preferred name and its definition; clicking on a concept in this panel makes all synonyms to appear in the right panel;

the bottom right panel displays the synonyms of the concepts that were selected either in the middle panel, or in the bottom right panel.
MetaCoDe is not a fully integrated tool yet, and it still requires some effort from the user to make it work on a new corpus. We hope that this guide will make this task easier.

It is still a simple tool. A lot of improvement are to be done, and may be a lot of them are even basic.

A good question is about the real benefit of the specific resource extraction step. It was wise to do so when machines had not so much memory as they have now; today this step may be achieved once for all during the installation step, rather than at tagging time.

The best way to improve a tool is to get feedback from people who tried using it. So we will be very pleased to receive any comment and suggestion at the following addresses:

  thierry.delbecque@sodad.com (preferred)

  or:

  thd@limsi.fr

Thanks.
Configuration Files

*extract_used_wd.param*

This is used for extracting the specific tables from UMLS before tagging. Its entries are the following ones:

- `dbi-dns` = [connect string to the database where UMLS resource is]
- `dbi-user-name` = [sql account to access this database]
- `pos` = [tree-tagger part-of-speech tags used to extract nominal phrases]
- `lang` = [language, en or fr]
- `vocabulary` = [vocabulary to use from UMLS, (if void, all vocabularies are used)]
This is used by the tag program. Its entries are the following ones:

- filein : [path to the file to process]
- cuisty : [path to the CUISTY file produced by extract-UMLS-resources.pl]
- wdsui : [path to the WDSUI file produced by extract-UMLS-resources.pl]
- suilength : [path to the SUILENGTH file produced by extract-UMLS-resources.pl]
- suicui : [path to the SUICUI file produced by extract-UMLS-resources.pl]
- suilimit : [upper bounds for SUI]
- cuilimit : [upper bounds for CUI]
- pospattern : [regular expression for nominal phrases identification]
- lempattern : [regular expression applied on lemmas for identifying nominal phrases]