Glossa: Administrators manual

Lars Nygaard, Anders Nøklestad

17th April 2008

Related documentation:

- GLOSSA User's Manual (doc/GLOSSA_manual.pdf)
- GLOSSA installation guide (INSTALL.txt)

0.1 A warning about terminology

For historical reasons, the word 'corpus' is used in a confusing manner in the Glossa source code and (some of the) documentation. It should be:

corpus a CWB corpus; may only be one language

project may comprise several CWB corpora (for multilingual corpora). In the future, one should log in to glossa/html/index_dev.php?*project*=XX not glossa/html/index_dev.php?*corpus*=XX

Chapter 1

Introduction

Technically, GLOSSA it is a front end for the CQP program, part of the IMS Corpus Workbench[1], the MySQL relational database[2], tgrep2[3] and Ted Pedersen's Ngrams Statistics Package[4].

A corpus instance:

- A html/php start page
- Configuration files
- CWB files
- Database tables:
  - bibliografic
  - sentences
  - lexical statistics
  - annotation

[1]
[2]
[3] http://tedlab.mit.edu/~dr/Tgrep2/
[4]

Chapter 2

Configuration

2.1 PHP/html

If you are using the standard PHP query page, you must edit the following regions:

```
if ( $_GET['corpus'] == 'test' ) {
echo "<script language='javascript' src='" . $htmlRoot . "/js/test.js'></script>";
}
--
if ( $_GET['corpus'] == 'test' ) {
include("test.inc");
}
--
if ( $_GET['corpus'] == 'test' ) {
include("test_cred.inc");
}
```

2.2 JavaScript

You need to create a javacript files called js/<projectname>.conf.js. So, for the monolingual project "nota", the following files is created:

```
var conf = new Array;
var languageOpts = new Array;
languageOpts = [['NOTA2', 'Norsk', 'no']];
var htmlRoot = 'http://omilia.uio.no/glossa/';
var cgiRoot = 'http://omilia.uio.no/cgi-bin/glossa/';
conf['type'] = 'monolingual';
var language='no';
```

For the multilingual "samno" project, we get:

```
var languageOpts = new Array;
languageOpts = [['SAMNO_SAMISK', 'Sami', 'sa'], ['SAMNO_NORSK', 'Norwegian', '
var conf = new Array;
conf['type'] = 'multilingual';
conf['title'] = 'Search Sami-Norwegian corpus';
conf['corpus_name'] = 'Sami-Norwegian Corpus';
var language='en';
var cgiRoot = 'http://omilia.uio.no/cgi-bin/glossa/';
```

Also, you might need to change the file js/dynamic_form_dev.js, around:

```
if (language == 'TEST') {
   reloadMenuTest();
}
```

2.2.1 Shortcuts

Keyboard shortcuts are defined in the <projectname>.conf.js file:

```
shortcut("Ctrl+Shift+L",function() {
   addOpt('w','lemma','grunnform');
});
```

'addOpt' can be used instead of clicking on the menu, and use the same values.

2.3 Configuration files

2.3.1 Main configuration

Below is an example of a corpus configuration file ("cgi.conf"):

```
db_pwd  =
db_name =
db_uname =
db_host = localhost
charset = UTF-8
charsetfrom = ISO-8859-10
htmlRoot = http://omilia.uio.no/glossa/
cgiRoot = http://omilia.uio.no/cgi-bin/glossa/
type = multilingual
logfile = /var/www/cgi-data/omclog
cwb_registry = /hf/omilia/site/corpora/cwb_reg
corpus_attributes = word lemma pos type grad_dia tense_defin case_mood
     person_type2 number gender polarity syntax
corpus_structures = s_id text_id
link_structure = text_id
diacr_table = /var/www/html/omc/diacr.dat
```

```
tmp_dir = /var/www/cgi-data/tmp
dat_files = /var/www/html/CE2/dat
download_url = http://omilia.uio.no/glossa/download/
config_dir = /hf/omilia/site/glossa-0.7/dat/
subcorp_files = /hf/omilia/site/glossa-0.7/dat/samno/subcorp/
hits_files = /hf/omilia/site/glossa-0.7/dat/samno/hits/
lang = en
meta_text = tid title publisher pubdate pubplace translation lang origlang
     tagger langvariety author translator classcode istrans
meta_class = class
```

The file format is simple: One entry per line; a keyword, a "="-sign, and a space separated list of values.

db_pwd,db_name,db_uname,db_host Login information for the metadata database

charset the character set used for displayed text

charsetfrom the character set used in the cwb-encoded data. If this is defined, the text will be converted from this character set to the one defined under *charset* before being displayed

htmlRoot the root directory for interface files (HTML, PHP and JavaScript)

cgiRoot the root directory for cgi scripts

type monolingual or multilingual

logfile full path to the query logfile

cwb_registry full path to the directory containing the registry file for the corpus

corpus_attributes the cwb attributes to be displayed (note that this is not necessarily the same as the attributes that are searched in).

corpus_structures the cwb structural tags to be displayed

link_structure (currently not used)

diacr_table conversion table for diacritics

tmp_dir directory where temporary search data is stored

dat_files where the other configuration files are stored

download_url URL from which the corpus can be downloaded(?)

config_dir location of the configuration files

subcorp_files location of stored subcorpus definitions

hits_files location of stored hits

lang the language used in the interface

meta_text the columns in the main metadata column ("text")

meta_class,meta_author the columns in the auxilliary metadata columns ("class" and "author")

groupfile the path to a file containing a space separated list of users that are allowed to access the corpus

charset the character set used in the interface (seen by the user)

charsetfrom if the charset of the CWB data, if it is different from 'charset' (typically used because CWB does not support Unicode).

2.3.2 Metadata configuration

The metadata interface is controlled by several configuration files. The main file, "meta.conf", is illustrated below:

```
collection      db      text    collection
title           db      text    title   where collection = ""
title-alle      db      text    title   where collection != ""
issnisbn        db      text    issnisbn
publisher       db      text    publisher
pubplace        db      text    pubplace
tid             db      text    tid
name            db      author  name    where in_collection = 1
name-alle       db      author  name    where in_collection = 0
geogr           file    author  geogr
geogr-alle      file    author  geogr
kategori        file    text    category
kategori-alle   file    text    category
emne            file    class   class
emne-alle       file    class   class
auth-type       file    author  type
auth-gender     file    author  sex
translated      file    text    istrans
```

It is a tab-separated file, where each line describes the content of a metadata widget. Each line has four mandatory columns and one optional column:

identifier the string used in HTML/PHP to create the widget.[1]

[1] All identifiers ending with "-alle" will be created as sub-menues if there exisists identifiers that is identical without this suffix.

type "db" or "file": where the program should fetch the content of the widget.

tablename which of the three tables ("text", "class", "author") the widget applies to.

column name which column the widget applies to.

constraint if only some of the (only applicable for "db"-type widgets).

If "db" is selected as type, the program extracts all possible values from the appropriate table and column (modulu the optional constrint) and populates the widget. For example, in the second line, the widget called "title" is populated with all the values from the column "title", in the table "text", for all entries where the "collection"-column is empty (in this case, removing newspaper articles etc).

If "file" is selected, the program reads the contents of the widget from the file named "<identifier>.dat":

```
original        n
translation     y
```

This is also a tab-separated file, where the displayed name of each entry is found in the first column, and the actual content (of the query to the database) is in the second column.

When the widgets are populated, they can be created in the HTML/PHP file of the corpus interface like this:

```
<script language="javascript">
  writeWidgetDoubleTable('title','tittel','hidden')
</script>
<br />
<script language="javascript">
</script>
<br />

<script language="javascript">
   writeWidgetCheck('translated','translated', 'open')
</script>
```

If your Glossa installation is going to handle multiple corpora, you would normally put the widget creation code for different corpora into different files (e.g. sami.inc, omc.inc etc.) and select which file to include at runtime depending on the selected corpus.

2.3.3 Menu generation

To generate a menu, you have to create a menu file, and run the create_menu_item.pl command. The menu file uses a sligthly idiosyncatic file format based on tab

separated fields:[2]

```
lemma   lemma form
case    case sensitive
start   start of word
end     end of word
middle  middle of word
neg     exclude
# w     word
word    add word
!word   add negated word
lemma   add lemma
!lemma  add negated lemma
# ADDSTRING     additional string
*       zero or more
+       one or more
?       zero or one
# occ   occurences
<break>
A       adjective
CC      CC
Pr      preposition
N       noun
Pron    pronoun
Po      postposition
V       verb
CS      CS
Adv     adverb
Interj  interjection
Num     numeral
Pcle    particle
# pos   Part of Speech
```

In the left column are the names as they are stored internally in the corpus, to the right are the names as displayed in the menu. Lines starting with a "#"-character designate categories; so when a user selects *"Part of Speech"* ▷ *adjective*, the internal query will be "[pos='adj']". A single line containing the string "<break>" will create a line break in the menu. A single line containing other words in angle brackets will create a heading in the menu.

The generation script is applied to this file to create a javascript file. This script takes three arguments, the corpus name, the nane of the javascript function, and the language in the menu (Norwegian or English). Typically, the last two parameters will be stored inside the program, so only the first one is needed:

[2]The format is likely to change in the future, when a standard format for all Glossa configuration files are established.

create_menu_item.pl <corpus-
name> <javascript_function> <language:en|no> < menu-
file > javascript-file
`create_menu_item.pl SAMI < sami_menu.txt > sami.js`

The Javascript file will typically be located in the Glossa directory, under /js.

Rule of Thumb: For maximum usability, try to find a balance between depth and breadth of the menu. In other words, neither the main menu or any of the submenues should contain more than about seven items. If long menues cannot be avoided, try to use line breaks or headings.

2.4 Tagset conversion

Because of a bug in CWB, there is a limit to how many positional attributes one may use.[3] Therefore, one may compress several non-overlapping categories in a single positional attribute, for example 'mood' and 'case' (which never apply to the same token). The name must contain an underscore.

The compressed categories can then be expanded with the file $GLOSSA/dat/<projectname>/multitags.dat. The format is (tab separated):

- name of compressed attribute (mood_case)
- a value (e.g. 'Nominative')
- the new attribute name (e.g. 'mood')

2.5 Files created by Glossa

Each search creates a set of results files, and various configuration files for the search. These are very useful for debugging. They are placed in a directory specified by the cgi.conf file, and start with the query id (can be found in the urls of the result page).

[3]The exact number depends on the lenght of the longest sentence in the corpus.

Chapter 3

Data used by Glossa

3.1 CWB files

For the creation of CWB registry and data files, you should refer to the CWB documentation. There are some extra things to consider, however:

- s_id should be unique over entire corpus
- tags should be converted into columns, where possible (jf. NLP::Tag2Cols)
- word forms and lemma forms should not contain spaces (use underscore instead)

3.2 Metadata database

The metadata database is optional, but has many possibilities.

All tablenames are prepended with the project name (in capital letters). Thus 'BOKMALtext' is the main table of the 'bokmal' project. The minimal text table contains three columns:

tid The text id used to connect other tables (must be the first column)

startpos The token number in the corpus of the first token in the text

endpos The token number in the corpus of the last token in the text

In addition, you can have all kinds of other columns (title, publisher, publication date etc.).

The token number is the line number in the cwb input file (not counting structural annotation: lines staring with '<'). You can extract start/stop positions from a CWB input file with the script $GLOSSA/bin/ positions_from_tab.pl

```
perl positions_from_tab.pl --mode=db --table=UPUStext --tag=text < cwbinput.txt
```

This will extract the position of the "text" structural annotation into the startpos/endpos columns of the UPUStext table.

You can create two additional metadata tables, for one-to-many relationships: author and class (named BOKMALauthor and BOKMALclass in this example). The first column of those tables must also be 'tid'.

3.2.1 Multiple start/stop-positions per text

For some corpora, particularly spoken language corpora, it's desirable to have multiple start/stop-positions. To do this, use a 'bounds_type=multiple' line in the configuration file, and populate the column 'bounds' with a tab-separated list of all the start-stop-positions (start/stop separated by a hyphen). Note that the positions must be in ascending order.

3.3 Lexical statistics

Glossa can access two kinds of sources for precompiled lexical statistics.

Firstly, the table called <projectname>_<corpusname>lexstat. This contains a column called 'form' and one column for each of the CWB positional attributes (lemma, POS etc.), and finally a column 'freq' with the frequency. The column will be used for the separate lexical statistics interface, and for the collocation interface (the 'Use global statistics' option, which will improve performance dramatically).

Secondly, a set of gzipped files, one for each text in the corpus, containing (at each line) lemma<tab>pos<tab>frequency. The name should be the 'tid' from the metadata table +"dat.gz". The files are used whan computing tables of words that are typical of a category (for terminology extraction etc.), in the lexical statistics interface.

3.4 Alignment table

For multilingual corpora, an alignment table is needed to show the clickable id to the left of the aligned corpus line. The table should be named <projectname>s_align, and contain

source sentence id

target sentence id

lang language (abbreviation)

3.5 Annotation

To allow user annotation of corpus positions, three tables must be created:

<projectname>annotation_sets Lists the various possible annotations.

<projectname>annotation_values Lists the various possibilities for the various sets.

<projectname>annotations User data about positions, using the sets and values of the other tables.

Chapter 4

Scripts for encoding data

The GLOSSA system does not care how you generate your data. But there are some utility scripts in the package that might come in handy.

4.1 Cwb data

TEI
- Simple text
- Tagged text
- Tagset conversion

4.1.1 Alignment

4.2 Metadata