Glossa: Administrators manual

Lars Nygaard

18th April 2007

Related documentation:

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

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

[1] `http://www.ims.uni-stuttgart.de/projekte/CorpusWorkbench`
[2] `http://mysql.com`
[3] http://tedlab.mit.edu/~dr/Tgrep2/
[4] `http://www.d.umn.edu/~tpederse/nsp.html`

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/<corpusname>.conf.js. So, for the monolingual corpus "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';
conf['title'] = 'Søk i Bokmålskorpuset';
conf['corpus_name'] = 'Korpus for moderne bokmål';
var language='no';
```

For the multilingual "samno" corpus, 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();
}
```

2.3 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
type = multilingual
logfile = /var/www/cgi-data/omclog
cwb_registry = /hf/omilia/site/corpora/cwb_reg
corpus_attributes = word lemma pos type degr_dia tense_defin mood_case \
     person_type2 number gender
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
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 Login information for the metadata database

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

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

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

2.3.2 Metadata configuration

The matadata 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]

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">
   writeWidgetFromTo('pubdate','utgivelsesår', 'hidden')
</script>
<br />

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

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

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
*       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:

create_menu_item.pl <corpus-
name> <javascript_function> <language:en|no> < menu-
file > javascript-file

[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 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 seven items. If long menues cannot be avoided, try to use line breaks or headings.

2.4 Tagset conversion

From tagger.

Categories (for display): "multitags.dat".

2.5 Files created by Glossa

Files created by users.

Log files.

Temporary search data.

Chapter 3

Data used by Glossa

3.1 CWB files

Reg
- cwb_data
- s_id should be unique over entire corpus

3.2 Metadata database

Tables
- Content

3.3 Lexical statistics

Tables
- Content
- stats/*.gs files

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

Appendix A

Terminology

"corpus": project name (bokmal, nota, test, omc, sami)
"base_corpus":
for monolingual:
- name of acutal files (ILN_LEKS, TEST, NOTA3)
- useful for versioning etc.
for multilingual:
- the corpus searched "first"
- the other corpora are aligned to it
- changes from query to query
"subcorpus":
- selection of texts (in the base corpus)