Autoglossing CHAT files with the Bangor autoglosser

Kevin Donnelly*

27 September 2010

Abstract

This manual explains how to use the Bangor autoglosser to provide glosses for CLAN[1] *.cha* files. One beneficial side-effect of this process is that the files are stored in a PostgreSQL[2] database, and this allows them to be used as input into the R statistical package[3] for detailed corpus-related queries.

1 Introduction

In the context of CLAN chat files, "glossing" means generating an additional tier containing part-of-speech tags. This autoglosser generates the gloss tier automatically from a POS-tagged dictionary. The software has been tested on Welsh and Spanish, and could be easily extended to handle any language, provided that you have available:

- a digital dictionary for that language, and
- a basic Constraint Grammar[4] to disambiguate homonyms in the language.

If you do not already have access to such resources, you will have to create them - some suggestions on this are given in later sections.

The Bangor autoglosser is licensed under version 3 of the GPL,[5], and was originally developed for use with ***Siarad***, the GPLed corpus of Welsh conversations assembled by the ESRC Centre for Research on Bilingualism in Theory

*kevin@dotmon.com
[1]http://childes.psy.cmu.edu/clan
[2]http://www.postgresql.org
[3]http://www.r-project.org
[4]http://beta.visl.sdu.dk/constraint_grammar.html
[5]http://www.gnu.org/licenses/gpl.html

and Practice,[6] at the University of Wales, Bangor. The software has been built and tested on GNU/Linux, but it is likely to run also on legacy platforms like Microsoft Windows or Apple Mac OS X.

2 Installation requirements

Your machine will need to have Apache2, PHP5, PostgreSQL, and (optionally) Git already installed. The Appendix gives a summary of the relevant commands for Ubuntu 9.10, and suggests how to configure your machine to use these applications effectively – note that this is only a suggestion, and other approaches are possible.

The Bangor autoglosser uses PHP scripts run at the command-line to import the text in CLAN chat files into a PostgreSQL.

3 Overview

The steps in the autoglossing pipeline are:

1. Prepare the file by removing broken lines (Section 6).
2. Import the file into a table (Section 8.
3. Remove non-word markup from the speech tier, and import the individual utterances into a table.
4. Look up each word in a language-specific dictionary to get a cohort of possible lemmas it could derive from, with appropriate part-of-speech tags.
5. Apply a language-specific constraint grammar to each cohort to remove inappropriate entries (i.e. to disambiguate the cohort).
6. Import the disambiguated entries into a table.
7. Combine information from the three tables to create a new text file where the original speech tier has a new gloss tier.

These steps can be combined by running one script(**do_autogloss.php**) to produce the glossed file automatically within a few minutes, or they can be accessed individually by running separate scripts for each step. This gives more flexibility should you wish to track down bugs, or make changes. For instance, if you have already glossed your file, but want to import it into a database for statistical or other analysis, you can run just two of the scripts, **cgimport.php** (Section 8) and **rewrite_utterances.php** (Section 10). Figure 1**??** summarises the scripts and the outputs at each stage.

[6]http://bilingualism.bangor.ac.uk

4 Putting a file through the pipeline

The autoglosser download contains a test chat file, which can be used to test the system. The test file is in the directory **inputs/testfiles**, and the default language is set as Welsh (see Section 5 below for more information on this).

Open a terminal in the autoglosser directory, and run:

```
php do_autogloss.php inputs/testfiles/mytest.cha
```

You should see text scrolling past on the screen, which is mainly feedback to let you know that the application is doing something.

If you examine the **autoglosser** database at the end of the process, you should find 3 new tables: **mytest_cgutterances** (containing each line of the speech tier), **mytest_cgwords** (containing the words in each speech tier line), and **mytest_cgfinished** (containing possible part-of-speech (POS) tags for each word).

In the **outputs** directory, there should be a subdirectory called **mytest**, containing output material from the autoglossing process. The two most important are **mytest_cg.txt** (the collection of candidate glosses that the constraint grammar will act on), and **mytest_cg_applied.txt** (the outcome after the constraint grammar has done its work).

Also in the output directory will be the end-product of the process, **mytest_autoglossed.txt**. This should contain a fully-glossed tier corresponding to the speech tier.

Notes:

1. It is advisable to have a standard location in the autoglosser filetree for the files that you wish to process, hence the **inputs** folder.
2. The file extensions currently catered for are **.cha** or **.txt**, but more can be added by editing the **get_filename()** function in the file **includes/fns.php**.
3. During the import, the extension is removed, and the name is lowercased, to be used as the *$filename* prefix for all tables and files created during the autogloss or import process. With the test file, the *$filename* is **mytest**. With an original file named Patagonia1, the *$filename* would be **patagonia1**.
4. The tables created at each stage of the process are given shorthand names: **$utterances** for the table *$filename_cgutterances*, **$words** for the table *$filename_cgwords*, and **$cgfinished** for the table *$filename_cgfinished*. This allows the names of the tables to be changed easily, should you wish to do so.

The following sections will look in more detail at what happens in the pipeline.

5 Setting the default language

IMPORTANT: This is an essential first step in importing transcriptions into the autoglosser. If this step is omitted, the files will not be imported correctly.

The Bangor autoglosser will import texts using old-style or new-style CLAN tags. Old-style tags attach numerals to every word, with the numeral designating the language, eg *gente@3*, where @3 designates Spanish. New-style tags use a language abbreviation attached to the word, eg *gente@s:es*. If you have a text using old-style tags, it is easiest to convert it to new-style tags before importing (although it is also possible to convert after importing), and a script is provided to allow you to do that (see Section **??**).

An important feature of the new-style tags is that one language is designated the default, and words from that language are not tagged. You therefore need to tell the autoglosser what language you are using as the default, or it will not look up the correct dictionary, and your output file will be deficient.

To do this, open the file **includes/fns.php**. Near the top of the file, there are some lines like this:

```
$zerolg=array("0");
$cylg=array("1", "cy");
$enlg=array("2", "en");
$eslg=array("3", "es", "");
```

The figures relate to the old-style tags, and the language abbreviations relate to the new-style tags. The important entry is the empty tag – `""` – which specifies the default language. In the above case, the default language is set to Spanish (es). If you wanted to set it to Welsh (cy), the lines should be edited to read as follows:

```
$zerolg=array("0");
$cylg=array("1", "cy", "");
$enlg=array("2", "en");
$eslg=array("3", "es");
```

In other words, just move the empty tag to the same line as your default language.

6 Prepare the file

The script **prepare_file.php** is a container for various processing tasks to be done to the file before import. Currently, it contains two sed[7] scripts:

[7] http://sed.sourceforge.net/

1. **utils/sed_joinlines**: The CLAN application seems to soft-wrap files at around the 80-character mark, so this sed script "straightens out" the lines so that each tier is on one unbroken line. *** Check whether there is a setting in CLAN to avoid this.
2. **utils/sed_convert_lgid**: This sed script converts old-style CLAN language identification tags (*@0, @1*) to new-style ones (*@s:en, @s:cy&es*) (see Section 5). This script is commented out by default (that is, it will not run unless you remove the // at the beginning of the line)

7 Running the scripts individually

You can run scripts individually on the same lines as you ran:

```
php do_autogloss.php inputs/testfiles/mytest.cha
```

The initial script of the sequence must refer to the CHAT file itself, including the path:

```
php prepare\_file.php inputs/testfiles/mynewfile.cha
```

Thereafter, the command can refer to the normalised name of the file (see Section 4, Note 3 above):

```
php write\_cohorts.php mynewfile
```

8 Import the file

The script **create_cgutterances.php** opens the CHAT file and imports it into the table **$filename_utterances**. The table structure is:

```
utterance_id integer NOT NULL,
filename character varying(50),
speaker character varying(10),
surface text,
[other tiers] text,
comment text,
durbegin integer,
durend integer,
duration character varying(50)
```

The CHAT file is scanned to discover additional tiers (eg %eng, %gls, %pho, etc). Columns for these will be generated on the fly in the [other tiers] location in the table.

The script **create_utterances.php** is called. This deletes any existing **$filename_utterances** table, and recreates it.

The speech tier is read into the column *surface*, and the sound data split off into the 3 end columns. Two functions are called to fix any obvious punctuation and transcription errors in the speech tier – these functions can be adjusted to suit yours own needs.

If there is a gloss tier (%gls) this is read in to a generated *gls* column as well. A gloss tier is unlikely in files that you are going to autogloss, but not impossible. For instance, you may wish to compare a manual gloss with the autogloss.

If there is an English interpretation tier (%eng), this is read in to a generated *eng* column. Again, this may not exist in all files.

Finally, comment lines (@Comment) are read in. These are not optimally-handled at the minute – we assume only one. In a series of consecutive comments, the last one will overwrite previous ones. FIXME: they need to be concatenated.

Output is fed to screen as the file is processed, and a log file is also written to disk (**$filename_cgutterances.txt**).

9 create_cgwords.php

This drops any existing $words table and recreates it. The table will hold the words from the utterance segmentation. *** Again, consider adding indexes. Again, change the welsh column name. We add sourcefile for tracking purposes, but glossloc is probably no longer necessary, since under the new system it will be the same as location, unless the person doing the glossing has made an error.

10 rewrite_utterances.php

Lifts the utterances out of the $utterances table and cleans them of any non-word items, eg markers, indicators, etc. This is in contrast to the previous system, where we tried to retain these. *** Is there an argument for retaining backtracking markers by appending them to the previous word? This might help in disambiguation, though it might also slow down the import a good bit. Add / to the allowed characters, and then replace space between word and / with an underscore?

The cleaning is done using the clean_utterances() function, which can of course be extended. Note that the order of the items in the function is significant - the main line removes anything that is not a letter or a few other things, so this has to be run after lines that remove items contained in square brackets. Also note that if you are using language tags containing & (the current CLAN recommendation), each of these has to be moved out of the way first, and then moved back again, because otherwise the lines getting rid of things like

&=laugh will affect them too: you will get party@scy instead of party@s:cy&en.

The original and cleaned utterances are written to a file ($utterances.txt) for checking purposes.

The speech tier is segmented at a wordspace, and written into the $words table. Note that the current language identification is based on the Siarad @ tag – this will need to be changed if working on other files. *** Would it be possible to abstract this into a function, so that different tagging systems could be called with a switch?

Any manual glosses are also written into the table. *** At present there is a check on whether they exist, but this may have no effect - needs to be tested.

11 write_cohorts.php

It would be sensible to allow this and the next file, apply_cg.php, to be callable standalone, because they need to be run multiple times when developing the constraint grammar. *** Add code that will handle arguments to the file, so that the $words tablename can be given directly. Unfortunately, this will not handle the function and dbconfig lines, so we need to duplicate those here, rather than relying on the ones in do_autogloss.php.

This lifts out all the surface words in the $words table and looks them up in the dictionaries. Then it writes the surface word and it's "cohort" of possible lemmas into a file in the CG format. See the tutorial on CG for more details.

The dictionary id of each word is also printed, to make it easier to make changes to the dictionary.

Which dictionary to use is based on the langid in the $words table, so this needs to be adjusted here. *** Perhaps use global variables to set language tags? Would also need to apply to dictionary table names.

In each case, if the word is unknown, it is given the UNK tag, but if it begins with a capital, it is given the NAME tag.

Output goes to screen, and also to a file, filename_cg.txt. *** Review this name.

12 apply_cg.php

This runs vislcg3 on the filename_cg.txt produced by the previous component, using the specified grammar. *** Again, this could be set somewhere as a variable, or handled as an argument to the file. On the other hand, having it in the file, at least during development, makes more sense.

The disambiguated output is written to a file, filename_cg_applied.txt.

13 write_cgfinished.php

The script create_cgfinished.php drops the table filename_cgfinished and recreates it.

Then it reads the disambiguated cohort file into this table. In cases where CG has been unable to reduce the cohort to one entry, glosses are concatenated, and separated by a slash.

It currently outputs a file, cgout.txt, but this is not very useful, since it shows concatenated glosses on separate lines.

14 write_cgautogloss.php

Currently, this is the last step in the autogloss procedure.

This collects information from the three tables generated, and writes out a file, filename_autoglossed.txt, giving the original utterance and its glossed equivalent. If manual glosses exist, they will be written out too. *** Need to include code to select this based on whether the manual gloss column is filled.

Note that this file is currently NOT a chafile - it is for checking purposes only. This script needs to be extended to write out the chafile itself. This would then need to be tested to ensure it is fully compatible with CLAN.

15 Further work

The header details of the chafile are ignored by the import. In fact, they need to be brought in to a separate table (in which case, having a separate table for comments would not be so bad). Alternatively, they could be written out to a separate header file, and rejoined as part of the final write – possibly better since it is simpler, and we are already generating a small thicket of working files anyway.

Appendix:
Configuring Ubuntu 9.10

REVISE: If we are not using a web interface, the info on setting up Apache is not required. Need to decide on CLI and/or web ...

These instructions should also work on Ubuntu 10.04. In either case, they assume a properly-working desktop with network access.

A Install relevant software

Install Apache (webserver), PHP5 (scripting system), and PostgreSQL (database), phpPgAdmin (browser interface to PostgreSQL), and (optionally) Git (versioning system) and pgAdminIII (desktop interface to PostgreSQL):

```
sudo apt-get install git-core, apache2, apache2-utils, libapache2-mod-php5,
php5, php5-cli, php-pear, php5-xcache, php5-pgsql, postgresql, phppgadmin,
pgadmin3
```

B Configure Apache

B.1 Configure a virtual host

```
sudo nano /etc/apache2/sites-available/autoglosser
```

Place the following in the file:

```
<VirtualHost *:80>
ServerName autoglosser
DocumentRoot /srv/www/autoglosser/public_html/
ErrorLog /srv/www/autoglosser/logs/error.log
CustomLog /srv/www/autoglosser/logs/access.log combined
</VirtualHost>
```

B.2 Tell the PC about the new virtual host

```
sudo nano /etc/hosts
```

Add the following line:

```
127.0.0.1 autoglosser
```

B.3 Enable the site and restart Apache

```
sudo a2ensite autoglosser
```

```
sudo /etc/init.d/apache2 restart
```

B.4 Give your normal user access to the `/srv` directory

```
sudo chown -R myuser.myuser /srv
```

B.5 Create a directory structure for the virtual host you set up earlier

```
mkdir -p /srv/www/autoglosser/public_html mkdir /srv/www/autoglosser/logs
```

B.6 Create a front-page for the virtual host

```
cd /srv/www/autoglosser/public_html
```

```
nano index.html
```

Enter the following:

```
<html>
<head>
<title>Autoglosser</title>
</head>
<body>
Front page for autoglosser virtual host.
</body>
</html>
```

If you now enter **autoglosser** in the address bar of your browser you should see a page reading *Front page for autoglosser virtual host*.

C Configure PHP

Check the configuration of the **php.ini** file:

```
sudo /etc/php5/apache2/php.ini
```

Set the following parameters as given:

```
max_execution_time = 300
memory_limit = 64M
register_globals = Off
magic_quotes_gpc = Off
magic_quotes_runtime = Off
safe_mode = Off.
```

; UNIX: ”/path1:/path2” include_path = ”.:/home/kevin/autoglosser/includes”

Restart Apache:

```
sudo /etc/init.d/apache2 restart
```

D Configure PostgreSQL

D.1 Set PostgreSQL to use passwords

```
sudo nano /etc/postgresql/8.4/main/pg_hba.conf
```

Change the line:

```
local all all ident
```

to:

```
local all all md5
```

D.2 Create a database user

```
sudo -i
```

```
su - postgresql
```

```
createuser -P mypguser
```

Enter a password (twice), and enter **y** at the superuser question.

Enter **exit** twice to return to your normal (desktop) user.

You will have to use the username/password you have just entered to replace the default *kevin/kevindbs* near the top of the `.php` scripts in the download.

E Download the autoglosser

In a working directory of your choice, run:

```
git clone http://thinkopen.co.uk/git/autoglosser
```

The files will be downloaded into a *autoglosser* folder in your working directory.

If you have chosen not to install Git, you can instead download the files by going to **http://thinkopen.co.uk/git/**, clicking on **autoglosser**, then on **tree**, and finally on **snapshot**. This will download a tarball containing the files. Uncompress this to create a `autoglosser` folder in your working directory.

Copy all the files in the `segmenter` folder to your new web directory at `/srv/www/segmenter/public_html`. Delete the **index.html** file you created earlier at B.6.

F Initialise the database

F.1 Create the database

Go the the `dbs` folder and create a new database:

```
createdb -U mypguser autoglosser
```

using the PostgreSQL user you created earlier at D.2. Enter your PostgreSQL password when prompted.

F.2 Import the tables

[FIX: This section may not be necessary, since the autoglosser creates the tables on the fly.]

Log in to psql as your user:

```
psql -U mypguser autoglosser
```

At the `autoglosser=#` prompt, enter:

```
\i table1.sql
```

```
\i table2.sql
```

Enter `\q` to exit **psql**.

F.3 Configure the database connection

Open a text editor, and replace the username and password in the file **autoglosser/config.php** with the PostgreSQL username and password you created earlier at D.2.

[FIX: All these files need to have **siarad** changed to **autoglosser**.]

Move the configuration details out of the web tree:

```
sudo mv autoglosser /opt
```

G Test access

Open a web browser, and enter **autoglosser** on the location line. You should see the front page of the autoglosser.

[FIX: Perhaps not required here - see above.]

If this does not work, contact me at **kevin@dotmon.com**.

H Ongoing database administration?

To administer PostgreSQL, it is easiest to use a frontend. You can use pgAdminIII[8], a desktop interface, or (preferably) phpPgAdmin[9], a browser-based interface. Open a browser and enter *localhost/phppgadmin* on the location bar. Enter your PostgreSQL username and password, and you should see a list of databases in the left-hand panel - since this is a new install, there are only two: the *postgres* system database, and *autoglosser*, the one you have just created. Click on *autoglosser*, and then on *Schemas*, and then on *Tables*. There are no tables available yet in the autoglosser database, because you have not created any - the import process will create 3 tables for each file you import.

I Notes

You need to copy the autoglosser/autoglosser dir to /opt, even if you are not using a website. Each page looks for the config.php in the /opt dir, so without this you will get errors when the scripts can't open the database.

Make inputs and outputs dirs in autoglosser, to give a launchpad for imports and a landing-pad for the outputs. Make symlinks in the inputs dir to your data dirs as necessary.

Download VISLCG3. Ensure build-essential is installed. Install libicu-dev. Untar the download, in that dir run sh autogen.sh, then make, then sudo make install.

Download ExPex zipped bundle from http://www.math.neu.edu/ling/tex/. Unzip. sudo cp -R expex/ /usr/share/texmf-live/tex/latex. sudo mktexlsr.

[8]http://whatever
[9]http://whatever

Install tofrodos via Synaptic. This gives fromdos, which is used in straightening the lines of the raw chat files so that they can be imported correctly.

Tenses, moods, aspects		Verbal extensions	
comp	completive **-sha-, -isha-, -kwisha-**	*assoc*	associative extension **-an-**
conc	concessional **-nga-**	*caus*	causative extension **-iz-, -ez-, -ish-, -esh-, -y-**
cond	conditional **-nge-**	*cont*	continuative extension **-t-**
cons	consecutive **-ka-**	*conv*	conversive extension **-u-, -o-**
curr	current present **-na-**	*inc*	inceptive extension **-p-**
fut	future **-ta-**	*pass*	passive extension **-w-**
hab	habitual **hu-**	*pos*	positional extension **-m-**
hypo	hypothetical **-japo-**	*prep*	prepositional extension **-i-, -e-**
imp	imperative **-e**	*stat*	stative extension **-ik-, -ek-**
gen	general present **-a-**		
part	participial **-ki-**		
past	past **-li-**		
perf	perfective **-me-**		
subj	subjunctive **-e**		
supp	suppositional **-ngali-**		

Table 1: *Tense and extension tags*

Kept temporarily as a table template.