Autoglossing CHAT files with the Bangor autoglosser

Kevin Donnelly[*]

28 May 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.

3 Overview

The stages in the autoglossing pipeline are:

1. Prepare the file by removing the headers and joining any broken lines (Section 4).
2. Import the file into a table (Section 5 - filename_cgutterances).
3. Remove non-word markup from the speech tier, and import the individual utterances into a table (filename_cgwords).
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 (filename_cgfinished).
7. Combine information from the three tables to create a new text file where the original speech tier has a new gloss tier.

4 Prepare the file

These steps are done manually, but can be automated later. Remove the header lines and paste them to **outputs/$filename.header**. Remove the **@End** line from the end of the file. Rename the file to **$filename_headerless**. Run:
`dos2unix inputs/$filename_headerless`

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

Replace filenames in **utils/sed_joinlines** and run:
`utils/sed_joinlines`
This will deal with Microsoft Windows endlines by ensuring that all lines of the text are joined up, and not split over several lines.

5 Put a file through the pipeline

File: do_autogloss.php Output tables: $filename_cgutterances, $filename_cgwords, $filename_cgfinished Output file:

This is the base file which calls all the other components. This means that you can run a single command on a file as prepared in Section 4, and the glossed file will be produced automatically within a few minutes. If you have already glossed your file, but want to import it into a database for statistical or other analysis, it is easiest to run the first two commands, cgimport.php (Section 6) and rewrite_utterances.php (Section 8) manually.

From a terminal, navigate to the location of the autoglosser, and enter:

```
php do_autogloss.php /path/to/mychafile.cha
```

adjusting the path as required for your setup, for instance:

```
php do_autogloss.php inputs/Patagonia1.cha
```

The file extensions currently catered for are **.cha** or **.txt**, but more can be added by editing the **get_filename()** function in **includes/fns.php**.

The extension is removed, and the name is lowercased, to be used as the **$filename** prefix for all tables and files created during the autogloss or import process. In the above example, the filename would be **patagonia1**. The tables that will be created at each stage of the process are given shorthand names: **$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 to so.

6 Import the file

- File: **cgimport.php**, also calling **create_cgutterances.php**
- Function: Opens the chat file given earlier, and imports it into the table **$filename_utterances**.
- Output tables: **$filename_cgutterances**
- Output file: none

We call the file create_cgutterances.php to drop (using the drop_table() function in includes/fns.php) any existing filename_utterances table, and recreates it. The table will hold the segmented chafile. *** Can change column names here to deal with gloss issue above. Maybe need to add indexes? Any benefits in adding another column which holds the complete speech tier, ie without splitting it up? This would simplify rebuilding. Need to change the table name from ”welsh”, so that other languages can be used. Also need to add a column for the autogloss, so that it can co-exist with any manual gloss.

We add the sourcefile as a column just in case it might be useful in doing queries. In cases where this is not available in the sound data, we use $filename generated earlier. *** Perhaps best to generalise this, and use it even with Siarad. *** sample_id and line_num may not be needed.

Read in the speech tier, beginning with *. Split off the sound data.

Read in the glosses, beginning with %gls. This may not exist in all files, and is likely not to in files that you are going to autogloss. Remove non-morphological strings that shouldn't be there (eg x, xx, xxx). *** Again, this should maybe be abstracted into a function, which could be added to as necessary. As noted above, perhaps change the column name to mygloss, or manual, or something, so that we have a clear run for the autogloss.

Read in the English interpretation, beginning with %eng. Again, this may not exist in all files. *** Need to find a way of handling any tier, eg %pho, %mor, etc.

Read in comments, beginning with @Comment. *** Comments are not well-handled – we assume only one. We either need to concat, as with the CG output, or write to another table linked to this one, using utterance_id as the key. The latter would mean two tables, and might be less easy to comprehend.

Output is fed to screen as we go through this process. *** Also write it to a file?

7 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.

8 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.

9 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.

10 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.

11 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.

12 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.

13 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[7], a desktop interface, or (preferably) phpPgAdmin[8], 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 4 tables for each file you import.

[7]http://whatever
[8]http://whatever

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*