Autoglossing CHAT files with the Bangor autoglosser

Kevin Donnelly

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[3] statistical package for detailed corpus-related queries.

1 General

The environment needs to be set up as in the earlier manual - ie PHP, PostgreSQL, etc, with a set location for the configuration information for the database. Apart from that, all that is needed is to put your chafile in the inputs dir, open a terminal, and run "php do_autogloss.php".

2 do_autogloss.php

This is the base file which calls the other components. This can be truncated to produce a file that only does imports (eg if you have already glossed your file). *** Need to think about whether we do a sample file like this. And also about the naming of the glosses column - perhaps assume autoglossing and keep gloss for that column, with mygloss for manual glosses. Need to find a way of writing all the output into a log - at present, opening a file in the script and writing to it doesn't work. *** May need to review filenames, tablenames, etc. Create folder for each file, to hold byproduct files?

Currently, the chafile is hardwired into the script, but it can be handled via an argument to a CLI script, or via a text-box on a webpage. CLI is probably preferable. *** Use these to allow subsets of the components to be selected?

It strips the extension .cha, and lowercases the name, to be used as the prefix for all tables and files created during the autogloss or import process. Two

[1] `http://childes.psy.cmu.edu/clan`
[2] `http://www.postgresql.org`
[3] `http://www.r-project.org`

variables are created that are used in the other components: $utterances (referring to the table filename_cgutterances) and $words (referring to the table filename_cgwords).

3 create_cgutterances.php

This drops any existing $utterances file (using the drop_table() function) 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 should use $filename generated earlier. *** sample_id and line_num may not be needed.

4 cgimport.php

Opens the chafile given earlier, and imports it into the table $utterances. This file could be tidied up a good bit. *** This was built to do Siarad, and we may need to do different flavours, even within projects. With the Patagonia files, for instance, the sound bullet holds only the duration; in Siarad it holds the string snd, and the name of the file. *** We need to include a joinline routine first, to deal with Microsoft Windows endlines. Sed should be OK for this.

First do some punctuation and error correction. These may need to be extended for other files. *** This could perhaps go into a function?

Read in the speech tier, beginning with *. Split off the sound data. *** This will not work in cases where snd is not used to preface this. Perhaps need to use the NAK character as the split-point.

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?

5 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.

6 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.

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.

===At this point the current pipeline stops, and we need to integrate subsequent files into it.===

7 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.

8 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.

9 create_cgfinished.php

This drops the table filename_cgfinished and recreates it. *** Put this into a separate file, like the other table-creation components above.

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.

10 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 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.

11 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.