Murrinhpatha corpus
A corpus focused on morphological data for a polysynthetic language

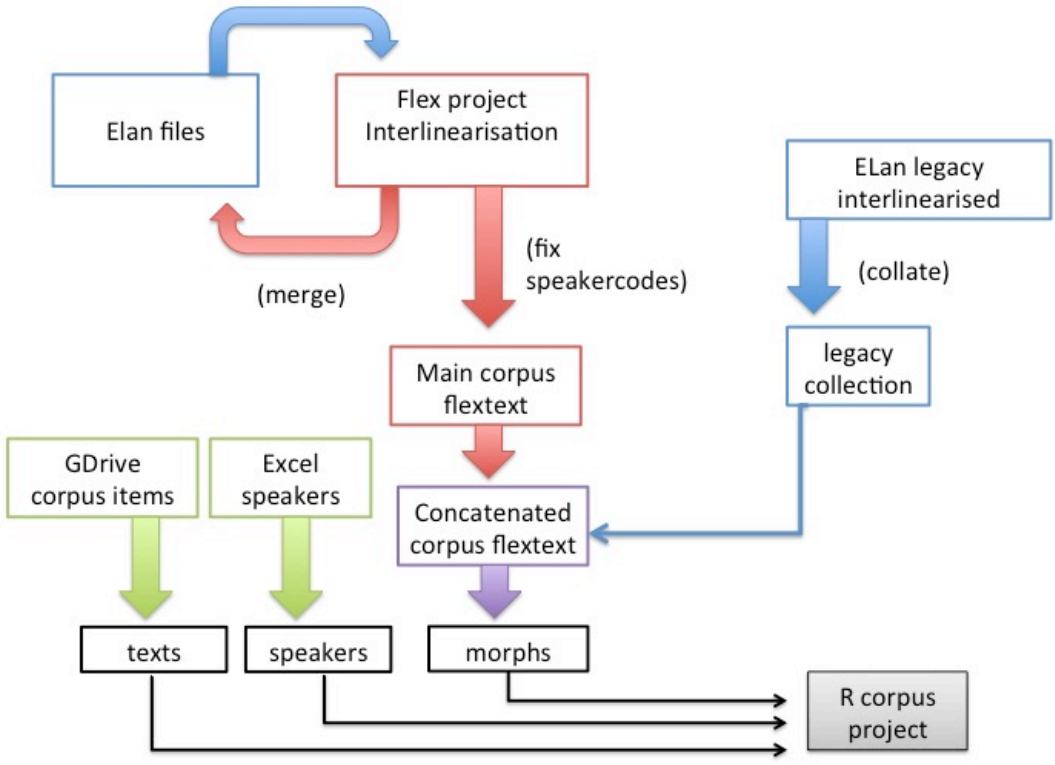
Workflow / data structure documentation

John Mansfield
11 April 2018

This document briefly describes some important features of the Murrinhpatha corpus data structure, and the workflow used to maintain it.

Earlier versions of the corpus had some different features, such as the use of a MySQL database. This doc describes the corpus as of April 2018.

The workflow is encapsulated in this figure, the parts of which are explained below:



Main data origination via Elan and Flex

Most of the corpus originates as Elan files, which naturally begin with no interlinearisation. They are exported to a Flex project for interlinearisation,via the .flextext format. After interlinearisation, it is exported from Flex, in the same .flextext format.

If the corpus text later requires further editing, additional transcription etc, then the interlinearised output from Flex is returned back into Elan for this purpose. This introduces an issue: if there are any 'custom' tiers from the original Elan transcript (i.e. anything other than phrases, translations and comments), these are lost in the Flex interlinearisation process. To make sure all data is maintained, the post-flex interlinearised version is merged with any custom tiers from the original transcript. This uses the script `merge-interlinear.xsl`.

For more detail on Elan/Flex cyclic workflow, see documentation here http://langwidj.org/linguistics/elan-flex-workflow/

Legacy interlinearised data

The Murrinhpatha corpus also has some older interlinearised data, which came from various Toolbox projects, rather than the current Flex project. This interlinearisesd data cannot be directly integrated into the Flex project. Attempts to import cause persistent errors, probably due to inconsistencies in glossing. However I integrate this data into the corpus by running it through a Python script that maps much of the legacy glossing onto the latest glossing conventions, and outputs it in .flextext format. Copies of all legacy texts are collated in a sub-folder within my `/database` folder. This uses scripts in the folder `/legacy-transform-data`, and also `collect-legacy-files.py`.

Concatenation, and transformation to tab-sep

The complete set of texts interlinearised in Flex are exported as a single .flextext file. As a first step after export, some speaker codes have to be fixed using `fix-speakercodes.xsl`, because these can't be edited in Flex. The collated set of legacy files are then concatenated with the main corpus file, using the script `concat-flexport-legacy.xsl`. This produces a single .flextext file containing all interlinearised data.

The complete corpus data is then transformed into a tab-sep file, where each morph is a record. Each morph is recorded along with the word and phrase it appears in, allomorph and underlying form, the text it appears in, the speaker who said it, and various other data.

Corpus metadata

The corpus metadata is stored in two spreadsheets. One is the list of corpus items, i.e. 'texts'. Some important details for each text are the genre (narrative, elicited etc) and the recording date. The master version of this is on Google Drive, so that an RA can update it. I export to Excel from there.

The speakers metadata is stored in an Excel spreadsheet. This has controlled lists that ensure speakers are associated with fixed sets of demographic characteristics

such as clan, main language spoken, gender, place of residence etc. Date of birth is a controlled date field.

Note that the age of speaker for any given corpus text is 'triangulated' between recording date and DOB. Thus speaker age is not stored in primary data, but is derived.

Both the texts list and the speakers list are exported as tab sep.

R project

The morphs and the metadata are now all in tab-sep files, which together form the basis for an R project. This facilitates quantitative analysis of the corpus. (It may prove useful at some stages to have indep tab-sep lists of 'words' and 'phrases', but I have no use for these right now.)

Because morphs are associated with speakers and texts, morphs can be associated with any metadata fields required via table merge functions.