The tools I had created as a hobby are on my github page (<https://github.com/jimregan/irish-asr-data>); I’m currently reorganising the scripts to keep dataset-specific scripts together, to better reflect their purpose, etc.

**Normalisation**

The initial set of recordings I had available to me were recordings of public domain texts; the only editions of these that I could use were in the pre-standard orthography (/orthographies), so normalisation has been important for these texts. Even with newer texts, although high quality spelling and grammar checking tools are available, they don’t seem to be as widely used as they ought to, so normalisation is still a requirement for these texts. As well as that, particularly in TG4 subtitles, where the English and Irish words sound alike (and the Irish word is not particularly common: ióga, súisí), it would be best to use the Irish word, if only to avoid feeding English to the LTS system to generate pronunciations.

In both cases, Kevin Scannell’s standardiser has been particularly useful. It works like a monotonic (here, meaning it does not reorder words) statistical machine translation system. On top of its output, I run a script that performs a set of transformations to the words, to filter out suggestions that don’t require manual checking (I check the remainder by listening to the audio).

As Wikisource has been my source for text (or, where the text was missing, provided the tools to proofread the text), I reused the normalisation lexicon there, although (as Irish is not in active enough use to have its own server) there are a few hoops to jump through to enable the facility. For the public domain texts, I then use the Wikisource version with automatic normalisation, rather than running my own scripts.

For other texts, I use this normalisation lexicon in combination with the variants file from the pronunciation dictionary, described below.

Similar to the normalisation scripts for TG4 subtitles, I have a pair of scripts to apply normalisations to the JSON used with finetuneas, so the replacement can be specific to a particular segment, and a diff tool for generating input to it from a pair of files, though there is a bug with insertions and deletions that I hope to get fixed in the next few days.

**Pronunciation dictionary**

The original set of seed words for the pronunciation dictionary was generated from the Teanglann Pronunciation Database (<https://www.teanglann.ie/en/fuaim/>), with predictable errors corrected (missing phones for words containing ‘z’, etc.)

The dictionary is split into three files, one per dialect, with an additional file (variants.tsv) that contains a variable number of fields: word, variant, dialect(s), and 1-3 fields for pronunciation, as relevant and in the order specified, so in

leo leofa CU l̻ʲ oː fˠ ə lʲ oː fˠ ə

field 4 is the Connacht pronunciation, and field 5 is the Ulster (in this particular case, the Ulster pronunciation was a later addition).

The first two fields are used in normalisation, so any instance of ‘leofa’ in the transcripts will be replaced with ‘leo’, to avoid ‘leofa’ being treated as an individual word, reducing perplexity.

The pronunciations from the variants file are added to the dialect specific dictionary, and they are concatenated together. For training with CMU Sphinx, the words in the dictionary must have a disambiguating number added to them, to allow the system to treat them individually (Kaldi adds these automatically: they are the disambiguation symbols #0#, #1#, etc.), and previous attempts to train CMU Sphinx have had lower accuracy because of this -- only the first variation of a word was used in training. To work around this for the Teanglann pronunciations, I intend to process the dictionaries and transcripts per dialect at the same time, noting the per dialect disambiguation number as words are added (the combined dictionary will work best without duplicate pronunciations) -- this idea only came to me while discussing it with Harald yesterday, so I haven’t yet had a chance to try it out.

English

I first attempted to generate pronunciations for English words by writing them in pseudo-Irish:

wellington ˈwɛlɪŋtən bhuelingtan vˠ e lʲ i ɲ tʲ ə n̻ˠ

But the sheer amount of words in the TG4 subtitles put me off doing this, as it is quite time consuming, and there is not always a reliable way of doing it.

The CMU pronunciation dictionary of English is multiple decades old, widely used, and well supported, so I tried to create a mapping to automatically transform it; there is an issue with this, as there is no separate phone symbol for schwa (it’s merged with ʌ). The naive approach of treating ‘u’ as ‘ʌ’ and schwa otherwise hasn’t quite paid off.

Today the thought struck me that using phonetisaurus to align graphemes to phonemes and then selecting ʌ or schwa would be more reliable; further to that, it would allow some Americanisms (e.g., a -> ɑ) to be replaced, which may make it more useful as a source for TTS.

Lexicon cleaning

The lexicon requires some cleaning, mostly due to errors that were only discovered much later: some were due to text processing errors, others due to not taking into account that compound words were generated in more than one part, etc.

CMU Sphinx (version 4) can incorporate a statistical phoneme to grapheme system, to try to generate a plausible transcription of unknown words, so it seemed ideal to use a statistical grapheme to phoneme system (the phoneme to grapheme model used by Sphinx is a grapheme to phoneme model in reverse), as correct output from both statistical and rule-based systems would match, but errors would not.

I had a false start with this by using Google’s festus system, which would have allowed me to reuse the same correspondences as the LTS rules (rather than relying on equivalents being learned automatically, as the other statistical grapheme to phoneme systems available do).

I had started to clean the abair lexicons of anything that could not have been generated by the LTS system to train an equivalent model using phonetisaurus. Having thought about editing phonetisaurus’s alignments for English, I think that generating an initial model without editing the lexicons, then filtering it to remove correspondences not allowed by the LTS rules, would save me that step -- it shouldn’t take more than a couple of hours to tell if it’s feasible, and if not, manually cleaning the lexicons is not too huge a task.