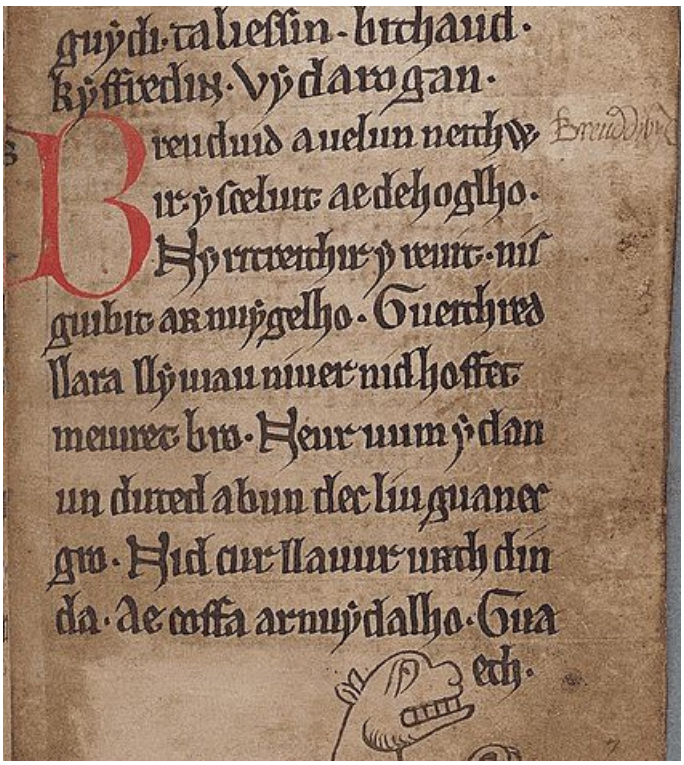
Autoglosser2

A glosser/tagger for Welsh



Kevin Donnelly

2016-18
(This version compiled 3 February 2018)

autoglosser.org.uk

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Being able to specify the part of speech of each word in running text (glossing or tagging) is the foundation for other natural language processing (NLP) tasks (examining word frequency or collocation, syntax analysis, machine translation, etc).

Autoglosser2 is a glosser/tagger for Welsh.[1] It is licensed under version 3 (or greater) of the Free Software Foundation's General Public License.[2] This means that, apart from costing nothing to use, it can be adapted and extended as required by the user, subject to the same license being used for any new version thus created and distributed.

Welsh is the most widely-spoken Celtic language, currently used by almost 20% (562,000)[3] of the population in Wales on a daily basis. Since the 1993 Welsh Language Act, it has been an official language in Wales, with public sector bodies required to give it equal status with English. Modern Welsh, due to its long contact with English, contains many English loans (conversation will usually include substantial amounts of code-switching), and its syntax has converged to some extent with English (e.g. periphrastic tenses have largely supplanted the native inflected tenses).

1.2 The Bangor Autoglosser

Autoglosser2 is a heavily revised version of the Bangor Autoglosser,[4] which was developed for Bangor University's then ESRC Centre for Research on Bilingualism in Theory and Practice in the period 2009-2011 (Donnelly et al., 2011; Donnelly and Deuchar, 2011a,b). That software, the first-ever dedicated Welsh glosser, was used to provide automatic glossing, based roughly on the proposals in Comrie et al. (2008), for the Centre's Welsh, Spanish and English bilingual conversational corpora, the results of which are available under a GPL license at the BangorTalk site.[5]

A detailed account of the development and research use so far of one of those corpora, the half-million word Welsh-English *Siarad* corpus, is available in Deuchar et al. (2018). Carter et al. (2017) and Carter et al. (2016) discuss the implications of using software such as the Bangor Autoglosser, while Deuchar et al. (2016) and Broersma et al. (2018) use data from the Autoglosser to draw significant new conclusions about the use of code-switching in Welsh.

1.3 Autoglosser2

A large part of the functionality of the Bangor Autoglosser was centred around handling conversational language which included code-switches (using English words and phrases in the Welsh, or switching entirely to English for some utterances). Autoglosser2, on the other hand, is aimed at written rather than spoken Welsh text, and has been refactored to tidy the code and make it far faster (it

[1] Proof-of-concept versions also exist for Gàidhlig (Scots Gaelic) and Māori.

[2] `gnu.org/licenses/gpl.html`. Components distributed with Autoglosser2 (HTML KickStart, Prototype, Jquery) have their own licenses.

[3] `statswales.gov.wales/Catalogue/Welsh-Language/WelshSpeakers-by-LocalAuthority-Gender-DetailedAgeGroups-2011Census`

[4] `github.com/donnekgit/autoglosser`

[5] `bangortalk.org.uk`

now glosses at a rate of 22,000 words/minute instead of 1,000). Like its predecessor, Autoglosser2 is rule-based – it uses constraint grammar to decide on the appropriate gloss or tag (Chapter 4). In this respect it differs from taggers which apply statistical methods to decide on the appropriate tag. The software again uses the Eurfa digital dictionary (Donnelly, 2016), the largest Welsh dictionary under a GPL license (Chapter 5).

Eurfa and some of the principles underlying Autoglosser2 (e.g. the use of constraint grammar) are also used in the CyTag software produced by the Corpus Cenedlaethol Cymraeg Cyfoes (CorCenCC – the National Corpus of Contemporary Welsh) project,[6] led by Cardiff University, although that tagger has been specially developed to handle the needs of the CorCenCC corpus. As well as the gloss format used by the Bangor ESRC corpora (Appendix B), Autoglosser2 is also able to output tags in CorCenCC format (Appendix C) – see Sections 2.2.3, 2.2.4, 3.9 and 3.10.

1.4 Getting started

Detailed instructions for installing Autoglosser2 and the other software it requires are in Appendix A.

Autoglosser2 has been developed on Ubuntu Linux 14.04, and tested on Ubuntu Linux 16.04, so it should run well on current versions of any Linux distro (though the instructions in Appendix A may need to be adjusted to take account of that distro's packaging approach). It may also run on legacy platforms like Microsoft Windows or Apple OSX, but is likely to need some tweaking – one way around this is to run Linux as a virtual machine on those platforms.[7]

A web interface is available at the Autoglosser2 website,[8] and included in the download (see Chapter 2), but this is intended only for short stretches of text, and for more demanding glossing (e.g. glossing a long file, or a directory of files) it will be best to use the command-line glosser (see Chapter 3), which is also likely to fit better into existing language processing workflows.

1.5 Contributing

Since Autoglosser2 is free software, you are welcome to change and extend it to suit your own requirements. Chapter 6 looks at areas where you might want to begin customising the output. I would be happy to take any suggestions on board – fork the repo, make your changes, and then submit a pull request, or send me your code directly (kevin@dotmon.com).

[6] `corcencc.org`
[7] `virtualbox.org`
[8] `autoglosser.org.uk`

Chapter 2

The Autoglosser2 web interface

2.1 Introduction

The quickest way to explore the Autoglosser2 is to use the web interface. If you have large amounts of text to convert, or you wish to integrate the glosser into a language processing workflow, you should use the command-line glosser – see Chapter 3. However, the web interface can be useful for testing shorter stretches of text when you wish to develop or fine-tune constraint grammar rules (Chapter 4) or want to check whether particular words are in Eurfa (Chapter 5).

An online instance of the web interface is available at `autoglosser.org.uk`. If you have followed the installation instructions in Appendix A/8, this interface will also be available to you on your own machine at the address *http://autoglosser2*. To use the interface, type or paste text into the input box.[1] Input is truncated to 300 characters, but if your text is longer than this you can tag it in chunks (see also Section 6.2).

Once your Welsh text is in the input box, press the button marked *Gloss it!*. This will segment the text into sentences, split each sentence into words, and begin the process of glossing (described in detail in Chapter 3). The output of the glossing process will be shown on the right-hand side of the browser window.

2.2 Output options

2.2.1 Default gloss output

Ticking the radio button *Glosses, horizontal layout* gives the default output, consisting of *Siarad*-type glosses (English lemma followed by POS tags) in a tiered format. For example, *droedio* would be tagged *tread.V.INFIN+sm*, meaning that it is a verbal infinitive (sometimes called a verbal noun), bearing soft mutation, and corresponding to the English lemma "tread". A full list of gloss components is given in Appendix A.

Figure 2.1: *Default web-interface output – horizontal glosses*

[1] Note that cutting and pasting from pdfs can often lead to an issue where the accent is pasted in as a separate glyph, and the "accented" letter, although it looks OK, will not be found on lookup. If you are find that accented words are being tagged as unknown, try deleting the "accented" letter from the input and typing it in manually.

Sample output (meaning “Lois is making a cake”) is given in Figure 2.1. Each input sentence is numbered, and below it in square brackets there is a gist (Wenglish) translation, which consists simply of a concatenation of the English lemmas for each word in the sentence. Below that there are two tiers giving the words in the text (with its location in the sentence marked via a superscript numeral), with the gloss below each.

Although not depicted here, in this and the other layouts any words that are undisambiguated (i.e. more than one part of speech could apply to them) or words that are unknown to Eurfa (i.e. need to be added to the dictionary) are marked in red.

A text file (ending in *_txtoutput.txt*) is also generated in the background, which may be useful for further processing. To access this, right-click the link at the top of the output area, and save the file. Note that if you click the link the file will open in a new browser window, but depending on the cache settings in your browser, the file that opens may not be the most recently-generated version, so right-clicking and saving is the best way to access the newest file.

The default output file is in a horizontal format, with four lines for each sentence: the (numbered) sentence, the gist translation, the tagged words, and an empty delimiter line – see Figure 2.2. Each tagged word consists of a number in braces giving the sentence and the word’s location in the sentence, the word, a hash symbol (#), and a gloss.

```
(1) Mae Lois yn gwneud cacen.
[is Lois  make cake . ]
{1,1}Mae#is.V.3S.PRES {1,2}Lois#Lois.NAME {1,3}yn#PRT {1,4}gwneud#make.V.INFIN
     {1,5}cacen#cake.N.F.SG {1,6}.#PUNC.FS
```

Figure 2.2: *Default web-interface output file – horizontal glosses.*

If there is no need to tag punctuation, this can be switched off in the web interface by ticking the box *In glosses, don’t tag punctuation*. Both web and file output will then filter out punctuation glosses, leaving only the punctuation mark itself.

2.2.2 Vertical gloss output

Ticking the radio button *Glosses, vertical layout* gives a variation on the default where the tagged words each appear on their own line, with the word preceded by sentence and location information in brackets, and followed by a hash and a gloss, again with an empty delimiter line between each sentence – see Figure 2.3.

(1) Mae Lois yn gwneud cacen.

[is Lois make cake .]

(1, 1) *Mae*#is.V.3S.PRES
(1, 2) *Lois*#Lois.NAME
(1, 3) *yn*#PRT
(1, 4) *gwneud*#make.V.INFIN
(1, 5) *cacen*#cake.N.F.SG
(1, 6) .#PUNC.FS

Figure 2.3: *Vertical glosses output.*

The equivalent output file follows the same pattern, except that the location information is in braces

– see Figure 2.4. As with the default layout, punctuation glossing can be switched off by ticking the box *In glosses, don't tag punctuation.*

```
(1) Mae Lois yn gwneud cacen.
[is Lois  make cake . ]
{1,1} Mae#is.V.3S.PRES
{1,2} Lois#Lois.NAME
{1,3} yn#PRT
{1,4} gwneud#make.V.INFIN
{1,5} cacen#cake.N.F.SG
{1,6} .#PUNC.FS
```

Figure 2.4: *Vertical glosses output file.*

2.2.3 Horizontal tag output

The Eurfa dictionary also includes CorCenCC-type tags for each entry, so ticking the radio button *CorCenCC tags, horizontal layout* will print CorCenCC tags instead of *Siarad*-type glosses for each word in the sentence. A full list of CorCenCC tags is given in Appendix C.

The web output (see Figure 2.5) each input sentence is numbered, with a gist translation below it in square brackets. Below that, each word in the sentence is preceded by its location, and followed by a hash and a tag.

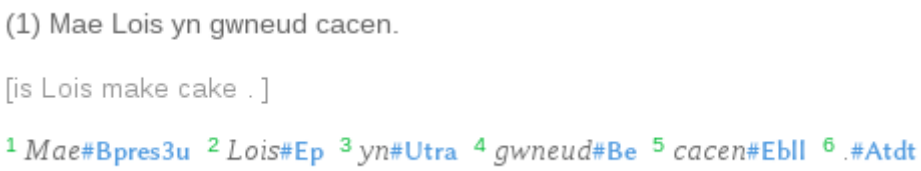


Figure 2.5: *Horizontal tags output.*

The equivalent output file is in a similar horizontal format, with four lines for each sentence: the (numbered) sentence, the gist translation, the tagged words, and an empty delimiter line – see Figure 2.6. Each tagged word consists of a number in braces giving the sentence and the word's location in the sentence, the word, a hash, and a tag.

```
(1) Mae Lois yn gwneud cacen.
[is Lois  make cake . ]
{1,1}Mae#Bpres3u {1,2}Lois#Ep {1,3}yn#Utra {1,4}gwneud#Be {1,5}cacen#Ebll {1,6}.#Atdt
```

Figure 2.6: *Horizontal tags output file.*

2.2.4 Vertical tag output

Ticking the radio button *CorCenCC tags, vertical layout* produces a vertical layout similar to the vertical gloss layout, but using CorCenCC tags. The tagged words each appear on their own line, with the word preceded by sentence and location information in brackets, and followed by a hash and a gloss, again with an empty delimiter line between each sentence – see Figure 2.7.

Figure 2.7: *Vertical tags output.*

The equivalent output file follows the same pattern, except that the location information is in braces – see Figure 2.8.

```
(1) Mae Lois yn gwneud cacen.
[is Lois  make cake . ]
{1,1} Mae#Bpres3u
{1,2} Lois#Ep
{1,3} yn#Utra
{1,4} gwneud#Be
{1,5} cacen#Ebll
{1,6} .#Atdt
```

Figure 2.8: *Vertical tags output file.*

Chapter 3

The Autoglosser2 pipeline

3.1 Introduction

This chapter explains the Autoglosser2 pipeline – a series of scripts that are run in succession to allow the import and glossing of your text. The examples will assume that you want to gloss or tag text in a file called `atext.txt` – wherever you see this, you can replace it with the name of your own file.

The commands here should be run in the command-line interface that is presented when you open a terminal (e.g. Konsole in KDE). They should be run from the top-level directory of Autoglosser2.

3.2 File format and location

The file you want to gloss or tag should be in plain text format – in other words, word-processor documents need to be converted to text format before import. The UTF-8 encoding is assumed, and if your file is in a legacy encoding such as ISO-8859 or variants, or Windows-1252, it may be wise to convert them first in case you get unexpected results during the glossing process.

For tidiness, you can place your input file in the `inputs` folder in the Autoglosser2 directory, but this is not essential – on first import you just need to give the full path to the input file's location. It is a good idea to keep the input filename lower-case and all-one-word. In contrast to Microsoft Windows, Llinux considers files with capitalised names as different files from the lower-case equivalent, and filenames containing spaces may not be handled as anticipated. If you need to include multiple words in the filename, link them with an underscore.

Note that where database tables are created, they will be created from scratch with each invocation of the script, with the old table being deleted. If for some reason you need to retain the old table (which is unlikely), you should back it up before running the script.

3.3 Import the text file into separate utterances

To import the file, run:

```
php import.php inputs/atext.txt
```
[1]

This will create a database table of the same name as your file, but ending in `_utterances` – in this case, `atext_utterances` – with each record in the table containing an utterance (which encompasses sentences, headings, list items, etc.) from your file.

The table contains four fields:

utterance_id A numeric identifier for the entry.
surface The Welsh text of the sentence.
translation An existing or generated translation for the sentence – left empty on initial import.
filename The name of the file containing the sentences – in this case, *atext*.

[1] If your input file is in some other location, just give its full path before the filename.

Using the example from Chapter 2, if the contents of `atext.txt` were *Mae Lois yn gwneud cacen.* ("Lois is making a cake."), the resulting contents of the `_utterances` table after import would be as shown in Table 3.1. As with all the tables produced by Autoglosser2, this table can be inspected using phpPGAdmin (Appendix A/12), SQLWorkbench/J (Appendix A/13), or any other database GUI.

utterance_id	surface	translation	filename
1	Mae Lois yn gwneud cacen.		atext

Table 3.1: *The `_utterances` table.*

The glossing process stores all output material (apart from the database tables) in the directory `outputs`, inside a sub-directory of the same name as your file. During import, two files are created in the background, and are therefore to be found in the directory `outputs/atext`. The first (`atext_sentencised.txt`) is a log file which holds the text at various stages in the import process – this allows the segmentation procedure to be reviewed and adjusted if necessary. The second (`atext_utterances.tsv`) contains the output of the segmentation, with the data items separated by tabs – the contents of this file are copied directly into the fields of the `_utterances` table. The tab-separated value (.tsv) format is used so that the file can be opened directly in a spreadsheet if desired, as well as being opened in a text editor.

3.4 Tokenise the utterances

To tokenise each utterance, run:

```
php wordify.php atext
```

Note that you only have to give the name of your file (without the *.txt* ending) – the script will use this to locate the `_utterances` table, read each utterance out of that, and split it into words (which encompasses punctuation, numbers, etc.).

This will create a database table of the same name as your file, but ending in `_words` – in this case, `atext_words` – with each record in the table containing a word from the file, ordered in terms of the sentence it occurs in.

The table contains 11 fields:

word_id A numeric identifier for the entry.

utterance_id The sentence in which the word occurs.

location The placement of the word in the sentence.

surface The Welsh word as it occurs in the sentence – the surface form.

lemma The base form of the Welsh word - the underlying form – left empty on initial import.

enlemma The base form of the English word corresponding to the Welsh word – left empty on initial import.

langid An identifier for the language of the word – left empty on initial import. This was an important element in Autoglosser1, which was designed to tag multiple languages at once, but is of less importance in Autoglosser2. The identifier uses the ISO-639-2 language codes,[2] or some other three-letter abbreviation (e.g. *han*, *cyr* – see below). Where the language is non-identifiable (e.g. in the case of punctuation), the identifier is replaced with three dashes.

auto The autogloss for the word – left empty on initial import.

corcencc The CorCenCC tag for the word – left empty on initial import.

filename The name of the file containing the sentences – in this case, *atext*.

semtag The semantic domain of the word – left empty on initial import. Not currently used in Autoglosser2, but retained in case of future development.

[2] `loc.gov/standards/iso639-2/php/code_list.php`

Table 3.2 shows the contents of the `_words` table after tokenisation. In this and subsequent tables, the fieldname *utterance_id* is abbreviated to *uid*, and location to *loc*.

word_id	uid	loc	surface	lemma	enlemma	auto	corcencc	filename
1	1	1	Mae					atext
2	1	2	Lois					atext
3	1	3	yn					atext
4	1	4	gwneud					atext
5	1	5	cacen					atext
6	1	6	.					atext

Table 3.2: *The `_words` table.*

During tokenisation, two files are created in the background, and stored in the directory `outputs/atext`. The first (`text_wordified.txt`) is a log file which holds the text at various stages in the tokenisation process – this allows the splitting procedure to be reviewed and adjusted if necessary. The second (`atext_words.tsv`) contains the output of the tokenisation, with the data items separated by tabs – the contents of this file are copied directly into the fields of the `_words` table.

3.5 Generate a constraint grammar cohorts file

Autoglosser2 uses constraint grammar (Chapter 4) to select the correct tag from multiple possible tags. This is done by applying the rules in a constraint grammar to a list ("cohort") of the possible tags available for a particular word ("readings").

To generate a cohorts file in a format that can be ingested by the constraint grammar parser, run:

```
php cohorts.php atext
```

Continuing with the above example, the resulting output, stored in the file `outputs/atext/atext_cg.txt`, would be as in Figure 3.1.

```
"<Mae>" atext,1,1
    "bod" [cym] v 3s pres :be: <B pres 3 u> + cap [59494]
    "bae" [cym] n m sg :bay: <E g u> + nm + cap  [196982]
"<Lois>" atext,1,2
    "Lois" [---] name :Lois: <E p> + cap
"<yn>" atext,1,3
    "yn" [cym] prt :: <U tra> [200654]
    "yn" [cym] prep :in: <Ar sym> [204430]
    "gan" [cym] prep :with: <Ar sym> + sm [196964]
"<gwneud>" atext,1,4
    "gwneud" [cym] v infin :do: <Be> [202152]
"<cacen>" atext,1,5
    "cacen" [cym] n f sg :cake: <E b ll> [209544]
"<.>" atext,1,6
    "." [---] punc fullstop :.: <Atd t> [214098]
```

Figure 3.1: *A sample constraint grammar cohorts file.*

The words in the sentence appear in quotes and angle brackets, followed by the filename and location information. Indented lines show relevant entries from Eurfa: the lemma in quotes, then the language identifier, then the part-of-speech information, then the English lemma, then the CorCenCC tag in angle brackets, then additional attributes such as capitalisation or mutation, and finally the number of the entry in Eurfa.

To create the `_cg` file, `cohorts.php` reads each word out of the `_words` table, and calls another file (`lookups/cym_lookup.php`) to look up that word in the Eurfa digital dictionary. The lookup covers

various aspects such as:

de-pluralisation e.g. *breichiau* → *braich* (“arms”), *canghennau* → *cangen* (“branches”)

de-capitalisation e.g. *Dirprwy* → *dirprwy* (“deputy”), *Cyd-Destunoli* → *cyd-destunoli* (“contextualisation“)

de-mutation e.g. *blant* → *plant* (“children”), *pharti* → *party* (“party”)

acronym resolution e.g. B. B. C. = B.B.C. = BBC

English recognition words that appear to be English will be marked as such, though no part of speech information will be given

glyph recognition e.g. 计划生育委员会 (*jīhuàshēngyù wěiyuánhuì* – “family planning committee”) will be identified as Han glyphs, Российская Федерация (*rossiyskaya federatsiya* – “Russian Federation”) will be identified as Cyrillic glyphs

basic URL recognition for addresses ending in *.com, .co.uk, .org, .org.uk, ac.uk*

number recognition e.g. *1980au* (decade), *137* (number), *18.12* (decimal), *VII* (roman numeral)

names any capitalised words that still have no entry in Eurfa are tentatively marked as names

During cohort generation, two files are created in the background alongside the main `_cg` file, and stored in the directory `outputs/atext`. They are intended to help the process of adding new items to Eurfa. The first (`atext_names.txt`) lists in alphabetical order the capitalised words that have been guessed as names, and the second (`atext_unknowns.txt`) lists in alphabetical order any non-capitalised words for which an entry has not been found in Eurfa. Both are often a useful way to pick up typos or misspellings in the original input text.

3.6 Apply the constraint grammar rules

To apply the constraint grammar to the generated cohorts file, run:

```
php apply_cg.php atext
```

The constraint grammar rules will prune the cohort of possible options for each word down to (hopefully!) a single correct option. The results will be stored in the file `outputs/atext/atext_cg_applied.txt`, and Figure 3.2 shows the sample text in Figure 3.1 after the constraint grammar rules have been applied.

```
"<Mae>" atext,1,1
    "bod" [cym] v 3s pres :is: <B pres 3 u> + cap [59494]
"<Lois>" atext,1,2
    "Lois" [---] name :Lois: <E p> + cap
"<yn>" atext,1,3
    "yn" [cym] prt :: <U tra> [200654]
"<gwneud>" atext,1,4
    "gwneud" [cym] v infin :make: <Be> [202152]
"<cacen>" atext,1,5
    "cacen" [cym] n f sg :cake: <E b ll> [209544]
"<.>" atext,1,6
    "." [---] punc fullstop :.: <Atd t> [214098]
```

Figure 3.2: *The results of applying constraint grammar rules to Figure 3.1.*

By default, two grammars are applied. The first (`grammar/cym_grammar.cg3`) is a general grammar, and the second (`grammar/cym_multiword.cg3`) marks words which are part of a multiword expression, noting their position in the multiword (left, middle, right). If you do not wish the multiword grammar to be applied, you can pass the option raw into `apply_cg.php`:

```
php apply_cg.php atext raw
```

If you are refining rules, it is helpful to be able to compare “before” and “after”. To enable this, `apply_cg.php` produces two other files giving the output from previous applications of the rules. The first

(atext_cg_applied_old.txt) contains the output from the last application, and the second (atext_cg_applied_old_old.txt) contains the output from the application before that. These files can then be compared in a difference viewer such as Meld[3] in order to check whether the rule changes have affected the desired word, and also whether there are any unexpected regressions relating to other words.

To investigate how rules are interacting with one another, you can also run:

```
php trace.php atext
```

This will produce a file – outputs/atext/atext_cg_traced.txt – which contains the results of the rule application in a format which shows all the cohort options Figure 3.1, but marked to show what rules were applied to arrive at Figure 3.2. The type of rule that has been applied is given at the end of the option, along with the line number of that rule in the grammar file, and the options pruned as a result of this are marked with a semicolon (;) at the beginning of the line – see Figure 3.3.

```
"<Mae>" atext,1,1
    "bod" [cym] v 3s pres :is: <B pres 3 u> + cap [59494] SUBSTITUTE:736
;   "bae" [cym] n m sg :bay: <E g u> + nm + cap [196982] REMOVE:155
"<Lois>" atext,1,2
    "Lois" [---] name :Lois: <E p> + cap
"<yn>" atext,1,3
    "yn" [cym] prt :: <U tra> [200654] SELECT:214
;   "gan" [cym] prep :with: <Ar sym> + sm [196964] REMOVE:138
;   "yn" [cym] prep :in: <Ar sym> [204430] SELECT:214
"<gwneud>" atext,1,4
    "gwneud" [cym] v infin :make: <Be> [202152] SUBSTITUTE:629
"<cacen>" atext,1,5
    "cacen" [cym] n f sg :cake: <E b ll> [209544]
"<.>" atext,1,6
    "." [---] punc fullstop :.: <Atd t> [214098]
```

Figure 3.3: *Tracing the application of constraint grammar rules to Figure 3.1.*

3.7 Gather part-of-speech data

Once the constraint grammar rules have been applied to the text so that (ideally) a single option has been selected for each word, the part-of-speech data for each word as given in the _cg.txt file is imported into a database table by running:

```
php cgfinished.php atext
```

This creates a database table of the same name as your file, but ending in _cgfinished – in this case, atext_cgfinished.

The table contains 15 fields:

id A numeric identifier for the entry.
filename The name of the file containing the sentences – in this case, *atext*.
utterance_id The sentence in which the word occurs.
location The placement of the word in the sentence.
surface The Welsh word as it occurs in the sentence – the surface form.
langid An identifier for the language of the word.
lemma The base form of the Welsh word - the underlying form.
enlemma The base form of the English word corresponding to the Welsh word.
auto The autogloss for the word. This consists of the POS data for each word, concatenated with a full stop.
dictid The number of the entry in Eurfa.
corcencc The CorCenCC tag for the word.

[3] meldmerge.org

mutation The type of mutation, if any, applied to the word.

cap Whether the word is capitalised in the text.

adjust Whether the word is part of a multiword expression, and if so, its location (left, middle, right) in the multiword. This field will not be used if you have elected not to apply the `cym_multiword.cg3` grammar – see Section 3.6.

semtag The semantic domain of the word.

Table 3.3 shows the main fields in the table once the data for the sample text has been imported.

id	uid	loc	surface	langid	lemma	enlemma	auto	corcencc	cap
1	1	1	Mae	cym	bod	is	v.3s.pres	Bpres3u	cap
2	1	2	Lois	—	Lois	Lois	name	Ep	cap
3	1	3	yn	cym	yn		prt	Utra	
4	1	4	gwneud	cym	gwneud	make	v.infin	Be	
5	1	5	cacen	cym	cacen	cake	n.f.sg	Ebll	
6	1	6	.	—	.	.	punc.fullstop	Atdt	

Table 3.3: *Part of the `_cgfinished` table, reflecting the application of the constraint grammar rules.*

During part-of-speech data collection, a `_cgfinished.tsv` file is created in the background, and stored in the directory `outputs/atext`. This contains the generation output, with the data items separated by tabs – the contents of this file are copied directly into the fields of the `_cgfinished` table.

3.8 Generate the glosses

To generate the glosses, the tags need to be uppercased and attached to the English lemma (strictly speaking, this step is unnecessary for the CorCenCC tags, because they are already in final form and do not need to have the lemma attached). But more importantly, if the word has not been disambiguated (that is, applying the constraint grammar rules have left more than one option for the word), the glosses and tags need to be combined so that they can highlight areas where the constraint grammar rules need to be refined.

To generate the glosses and any undisambiguated combinations, run:

```
php join_tags.php atext
```

This creates a database table of the same name as your file, but ending in `_holding` – in this case, `atext_holding`.

The table contains 10 fields:

id A numeric identifier for the entry.

filename The name of the file containing the sentences – in this case, *atext*.

utterance_id The sentence in which the word occurs.

location The placement of the word in the sentence.

lemma The base form of the Welsh word - the underlying form.

enlemma The base form of the English word corresponding to the Welsh word.

langid An identifier for the language of the word.

auto The autogloss for the word. This consists of the POS data for each word, concatenated with a full stop.

corcencc The CorCenCC tag for the word.

semtag The semantic domain of the word.

Table 3.4 shows the main fields in the table once the glosses have been generated. The generation process also includes additional tweaking of the data: for instance, mutation marking is attached to

the gloss; capitalisation is restored to nouns, adjectives, and infinitives; glosses for secondary words in multiword expressions are replaced with an arrow pointing towards the primary word. etc.

id	uid	loc	lemma	enlemma	langid	auto	corcencc
1	1	1	bod	is	cym	is.V.3S.PRES	Bpres3u
2	1	2	Lois	Lois	—	Lois.NAME	Ep
3	1	3	yn		cym	PRT	Utra
4	1	4	gwneud	make	cym	make.V.INFIN	Be
5	1	5	cacen	cake	cym	cake.N.F.SG	Ebll
6	1	6	.	.	—	PUNC.FULLSTOP	Atdt

Table 3.4: *Part of the `_holding` table, showing generated glosses.*

During tag generation, a `_joined.tsv` file is created in the background, and stored in the directory `outputs/atext`. This contains the gloss and combination output, with the data items separated by tabs – the contents of this file are copied directly into the fields of the `_holding` table. The relevant data is then transferred to the `_words` table.

3.9 Create pdf output

Once all the words in the text have been glossed, running:

```
php pdfoutput.php atext
```

will use the data in the `_utterances` and `_words` tables to generate a *.tex* file (in this case, `atext.tex`), which can then be processed by the LaTeX typesetting system to create a pdf (`atext.tex`) of the input text, in a tiered format. The original text is on the top tier, and below it is a tier containing the glosses – see Figure 3.4.

Figure 3.4: *Default pdf output.*

The default output in Figure 3.4. can be adjusted in a number of ways by passing various options (in any order, separated by `+`) to `pdfoutput.php`:

corcencc Display CorCenCC tags instead of *Siarad*-style glosses.

both Display both glosses and tags on separate tiers.

colour Display glosses or tags in colour. The colours can be specified by editing the pdfoutput.php script.

nopunc Suppress the display of glosses for punctuation marks.

notrans Suppress the translation line.

Multiple options can be passed in separated by a plus sign (`+`). Figure Figure 3.5 shows the output from running:

```
php pdfoutput.php atext corcencc+notrans
```

and Figure 3.6 shows the output from running:

```
php pdfoutput.php atext both+colour+nopunc
```

(1) **Mae Lois yn gwneud cacen .**
Bpres3u Ep Utra Be Ebll Atdt

Figure 3.5: *Options: corcencc + notrans.*

(1) **Mae Lois yn gwneud cacen .**
is.V.3S.PRES Lois.NAME PRT make.V.INFIN cake.N.F.SG
Bpres3u Ep Utra Be Ebll Atdt

is Lois make cake .

Figure 3.6: *Options: both + colour + nopunc.*

3.10 Create text output

Plain text output identical to that discussed in Chapter 2 can also be generated. The default output is a horizontal gloss – see Section 2.2.1 – but three other layouts are also possible:

agvertical Vertical gloss format – see Section 2.2.2.
cchorizontal Horizontal format using CorCenCC tags – see Section 2.2.3.
ccvertical Vertical format using CorCenCC tags – see Section 2.2.4.

The output file (`atext_txtoutput.txt`) can be adjusted by passing one of these options into the script – for instance:

```
php txtoutput.php atext agvertical
```

will produce a text file in the vertical gloss format.

3.11 Glossing speed

Glossing part of the CorCenCC Gold Corpus – a 3,075-word extract from the 9m-word Kynulliad3 (Donnelly, 2013) – gives a rough indication of the speed of the pipeline. Table 3.5 shows the average of three timings at the various stages of the pipeline when run on a 2014 laptop with an Intel i7-4500 CPU and 4Gb RAM.

Stage	Script	Duration
Import sentences	import.php	0.391
Tokenise sentences	wordify.php	0.346
Generate cohorts file	cohorts.php	2.841
Apply constraint grammar	apply_cg.php	0.257
Trace constraint grammar	trace.php	0.216
Collect part-of-speech data	cgfinished.php	0.408
Generate glosses	join_tags.php	0.456
Create pdf output	pdfoutput.php	3.181
Create text output	txtoutput.php	0.166
Total duration		**8.262**

Table 3.5: *Duration (in seconds) at the various stages of the glossing pipeline.*

Lookup and pdf generation are the slowest sections of the pipeline. Overall, the average glossing speed is 372 words/second, or over 22,000 words/minute.[4]

[4] This is a considerable improvement on the speed (1,000 words/minute) of the Bangor Autoglosser.

3.12 Running the entire pipeline

To avoid having to run nine scripts one after the other, a compendium script (`do_everything.php`) is provided to do this. Running:

```
php do_everything.php inputs/atext.txt
```

will run each script in succession, and generate all the outputs files and tables. The text to be imported should be given as the argument to the script.

The options described above – see Sections 3.6, 3.9 and 3.10 — can also be passed (in any order, separated by `+`) to `do_everything.php`. Thus:

```
php do_everything.php inputs/atext.txt raw+both+nopunc
```

will use raw tags for multiwords, and print a pdf file with both *Siarad*-type glosses and CorCenCC tags, skipping the glossing of punctuation.

```
php do_everything.php inputs/atext.txt corcencc+colour+notrans+ccvertical
```

will print a pdf file with only CorCenCC tags, displaying these in colour, and suppressing the gist translation, and will print a txt file in vertical format.

You can of course choose to have glosses in one file and tags in another. For instance:

```
php do_everything.php inputs/atext.txt colour+cchorizontal
```

will produce a pdf file with coloured glosses, and a txt file with horizontal tags.

3.13 Glossing a directory of files

If you have a set of files in a directory, you can avoid having to run `do_everything.php` on them individually by using the shell script `run_do_everything`.[5] You need to edit the script before running it. Your files should be placed in a directory in `inputs`, and the name of that directory should be specified in the *FILES* line. The outputs will be placed in a directory in `outputs`, and the name of that directory should be specified in the *corpus* line. You can then run the script:

```
./run_do_everything
```

[5] This is unlikely to work on Microsoft Windows without installing additional software, and may not work on Apple OSX.

Chapter 4

Constraint grammar

4.1 Introduction

> "Constraint Grammar (CG), launched by Fred Karlsson in 1990, is a methodological paradigm for natural language processing (NLP). Context-dependent rules written by a human linguist are compiled into a grammar that assigns grammatical tags to words or other tokens in running text.
>
> "Typical tags address lemmatisation (lexeme or base form), inflexion, derivation, syntactic function, dependency, valency, case roles, semantic type etc. Each rule adds, removes, selects or replaces a tag or a set of grammatical tags in a given sentence context."
>
> From: `en.wikipedia.org/wiki/Constraint_Grammar` (slightly edited)

Different versions of the constraint grammar formalism have been developed over the years. Karlsson's version (Karlsson, 1990; Karlsson et al., 1995) is now referred to as CG-1, and a later version that made some changes was known as CG-2.

The version of constraint grammar used by Autoglosser2 is the CG-3 variant, developed as part of the University of Southern Denmark's Visual Interactive Syntax Learning project[1] by Eckhard Bick and Tino Didriksen (Bick, 2009; Bick and Didriksen, 2015; Didriksen, 2017). CG-3 is the most featureful and versatile variant, and the only one under current development. It is free software, licensed under the GPL, and is being used in a number of NLP projects, including Apertium, a free (GPL) machine translation system.[2]

4.2 Key attributes of CG-3

The CG-3 constraint grammar parser applies rules in a "grammar" file (in Autoglosser2, `grammar/cym_grammar.cg3`) to text input in a specific format (in Autoglosser2, the file `outputs/atext/atext_cg.txt` created by `cohorts.php`). The rules are applied sequentially to each cohort of POS options (readings) for a particular word until only one candidate reading is left, and can be applied in batches if desired. The rules choose (select), remove, or add morphological, syntactic, semantic or other readings in particular contexts conditioned by surrounding words or tags. A particular strength of constraint grammar is that it can assign readings even to unconventional (non-standard) language such as the conversations in the Bangor ESRC corpora, and can handle codeswitching well (Donnelly and Deuchar, 2011b). Disambiguation (reduction of the readings to one correct POS option) regularly reaches 99%, and impressive results can be obtained with a grammar of as few as 200 rules. Grammar size can vary substantially, depending on language and tag complement – one French grammar has 1,400 rules, while a Danish grammar has 8,000.

A rules-based system like constraint grammar differs significantly from the current focus on machine learning, where the idea is that if a computer can be given enough text it will be able to increasingly accurately extract such things as tags. Rule-based systems tend to require linguistic knowledge rather than heuristic data-analysis – for this reason, they may take longer to develop, but may be the only viable option where large amounts of digital text are not available (for instance, in the case of minority languages). They are also likely to give more accurate results over different language registers, and do

[1] `visl.sdu.dk`
[2] `apertium.org`

not suffer from the possibility of implicit bias, where racial, gender and other biases in linguistic corpora carry over into machine learning output (Caliskan-Islam et al., 2017).

4.3 Constraint grammar rules

Donnelly (2010) gives a general introduction to constraint grammar, and further details can be found in Bick (2009), Bick and Didriksen (2015) and Didriksen (2017). This section will give a short overview of the grammar file `grammar/cym_grammar.cg3`, so that users can experiment with writing their own rules or amending existing ones.

4.3.1 Preamble

At the head of the grammar file, the *DELIMITERS* command specifies the items considered to mark the end of an utterance (sentence). This is followed by a group of *LIST* and *SET* definitions, which create sets and combine them respectively. These can be used to simplify the writing of the rules, but not all of them are used in the remainder of the file.

4.3.2 Rule sections

The rules begin after the *SECTION* command. There can be multiple rule sections, and they can be run sequentially, or in isolation, or in repeated groups. Possible reasons for multiple sections might be to separate ”safe” rules (to be used earlier) from rules of thumb (to be used later), or to separate rules with different areas of application (for instance, rules that act on one language in the text from those that act on another language). This grammar file currently has two sections: the main one and an experimental section near the end which adjusts the lemmas.

4.3.3 Rule application

Rules are always applied in the order in which they occur in the grammar file, and any given rule in a section will be applied to all cohorts in the sentence before moving on to the next rule in that section. Each section is applied iteratively.

A rule can act on any tag anywhere in a cohort reading – in other words, tags do not need to be in a particular sequence or order. Each rule is run against each cohort, to see if there is a reading in the cohort to which it should be applied.

Each rule ends with a semicolon (;), and begins with a keyword such as *SELECT*, *REMOVE* or *SUBSTITUTE*. By convention, these are written in upper-case, but they also work if written in lower-case, and this grammar file uses that option.[3] There are many other keywords, but these three are the only ones used in this grammar file.

4.3.4 Rule targets

Following the keyword is the *TARGET*, i.e. the word on which the rule will act. Targets can be specified as sets, or as tags, lemmas (enclosed in double quotes), or surface words (enclosed in angle brackets and double quotes). The last three must also be enclosed in parentheses. So:

```
select smtrigger if ...
```

would act on any words in the set of words that has been defined as triggering soft mutation.

```
select (n f) if ...
```

would act on any words which are tagged as feminine nouns.

[3] I find the rules easier to read in lower-case, but others may disagree.

```
select ("coes") if ...
```
would act on any word whose lemma is *coes*, i.e. *coes*, *goes*, *nghoes*, *coesau*, *goesau*, *nghoesau*, etc.
```
select ("<goes>") if ...
```
would act on any instance of *goes*, but not on related forms such as *coes*, *nghoesau*, etc. Note that the *if* is included in this file for readability, but is optional – the rules will work without it.

4.3.5 Rule contexts

The target can be followed by a set of contexts or conditions under which the rule will apply. If more than one context is specified, all of them must be met (instantiated) before the rule will apply. Contexts must be individually enclosed in parentheses, and usually consist of a position marker and a tagset. A positive position marker refers to text to the right (coming after the target), while a negative position marker refers to text to the left (coming before the target). A context can be negated by using *NOT* in front of the position marker.

If no context is specified, the rule will apply globally – this can be used to suppress rarely-occurring reading. For instance, the following rule will delete all readings containing a subjunctive verb, no matter what context they occur in:
```
remove (v subj);
```
This can also be used as a shortcut to select one meaning of a word. For instance *ysgol* can mean “school” and also “ladder”, but the first meaning is much more frequent. Ideally, semantic context would be used to determine which reading to choose, but until that is possible a rule can be defined that will always choose the ”school” meaning of the *ysgol* lemma:[4]
```
select ("ysgol" :school:);
```
or ban the ”ladder” meaning:
```
remove ("ysgol" :ladder:);
```[5]
This sort of global rule is used in the first few parts of this grammar file, to select most frequent lemmas or delete rare ones, and also to remove incorrect readings, e.g. incorrect depluralisations (such as *llewys*, “sleeves” being read as a plural of *llew*, “lion”), or impossible mutations (such as *na*, “than”, being read as a nasally-mutated form of *da*, “good”),

4.4 Some rule examples

4.4.1 SELECT rules

`select ("a" conj) if (-1 (name)) (1 (name));`
Select the reading of *a* (and) as a conjunction if the word to the left (*-1*) and the word to the right (*1*) are both names, as in *Conwy a Gwynedd* (Conwy and Gwynedd).

`select ("a" pron.rel) if (1 (subj));`
Select the reading of *a* (who, which) as a relative pronoun if the following word is in the subjunctive tense, as in *doed a ddelo* (come what may).

`select ("a" prt.int) if (1 (pres.indef));`
Select the reading of *a* as an interrogative particle if the following word is in a present indefinite verb, as in *a oes heddwch?* (is there peace?).

`select (prt.aff) if (not -1 (prep)) (1 inflected);`
Select the reading of *mi* as an affirmative particle if the following word is an inflected verb, but not if the preceding word is a preposition. This will apply to *mi welodd* (he saw), but not to *wrth i mi gyrraedd* (as I arrived).

`select ("mi" pron) if (-1 ("i" prep));`

[4] Obviously, this will give an incorrect gloss when the “ladder” meaning is actually intended!
[5] Note that in the output from `cohorts.php`, the English lemma is surrounded by colons, and they need to be replicated in the rule.

Select the reading of *mi* (me) as a pronoun if the preceding word is the lemma *i* used as a preposition, as in *i mi* (to me).

```
select ("â" conj) if (-1 (adj.eq));
```
Select the reading of *â* (as) as a conjunction if the preceding word is an equative adjective, as in *cyn belled â* (as far as).

```
select ("â" prep) if (1 (det.def) or (n) or (name)) (not -1 (adj.eq));
```
Select the reading of *â* (with) as a preposition if it is preceded by the definite article, a noun or a name, and not followed by an equative adjective, as in *aeth taid â'r ddwy ferch* (grandfather went with both girls) or *ffinio â Lloegr* (bordering [with] England).

```
select ("yn" prt) if (1C (adj));
```
Select the reading of *yn* as a particle if the following words is definitely (*C* for certain or careful) and adjective, as in *yn hapus* (happy).

```
select ("yn" prep) if (1 (det.def));
```
Select the reading of *yn* (in) as a preposition if the following word is a definite article, as in *yn y tŷ* (in the house).

```
select ("o" :of:) if (1 (det.def) or (adj.poss));
```
Select the reading of *o* (of, from) as a preposition if the following word is a definite article or a possessive adjective, as in *o'r môr* (from the sea) or *o'n ffrindiau* (from our friends).

```
select ("fo" pron) if (-1 (infin)) (1 (prt)) (2 (adj));
```
Select the reading of *o* (< *fo*, "he, him") as a pronoun if the preceding word is a verbal infinitive, the following word is the particle *yn*, and the word after that is an adjective, as in *i wneud o'n saff* (to make it/him safe).

```
select ("cyn" prep) if (not 1 (adj.eq));
```
Select the reading of *cyn* (before) as a preposition if the following word is not an equative adjective, as in *cyn mynd* (before going).

```
select ("deg" num) if (1 (n));
```
Select the reading of *deg* (ten) as a numeral if the following word is a noun, as in *deg awr* (ten hours).

```
select ("pryd" :meal:) if (1* ("bwyd"));
```
Select the reading of *pryd* with the English lemma "meal" if the lemma *bwyd* (food) follows (*) anywhere, as in *pryd o fwyd* (a meal).

```
select ("pryd" :time:) if (-1 ("pa" :which:));
```
Select the reading of *pryd* with the English lemma "time" if preceded by the lemma *pa* with the English lemma "which", as in *pa bryd?* (when?).

```
select ([eng]) if (-1 ([eng])) (1 ([eng]));;
```
Select the English reading if the word is preceded and followed by English words.

4.4.2 REMOVE rules

My suggestion is that *REMOVE* rules should be avoided unless the removal context is bullet-proof – they can have unintended consequences if the context is not carefully defined, and in most cases *SELECT* rules are to be preferred.

```
remove ("yn" prep) if (1 (sm));
```
Remove the reading of *yn* (in) as a preposition if it is followed by a word showing soft mutation – the preposition *yn* is followed by nasal mutation.

```
remove (prt.aff) if (-1 inflected);
```
Remove the reading of *mi* (in) as an affirmative particle if it is preceded by an inflected verb, as in *sgiwsiwch fi* (excuse me).

`remove ([eng]) if (0 ([cym]));`
Remove an English reading if a Welsh one exists for the current word (*0*).

4.4.3 SUBSTITUTE rules

Rules applying a substitution have four parameters: the items that are to be changed, the items they are to be changed to, the context in which those items occur, and the context in which they should be changed. To paraphrase: replace A with B in the context X when conditions Y apply.[6] Autoglosser2 is currently only using SUBSTITUTE rules experimentally to adjust the English lemma in the glosses.

`substitute (:class:) (:district:) ("dosbarth" [cym] n m sg :class:) if (-1 ("cyngor"));`
Replace the English lemma "class" with "district" when the lemma is *dosbarth* and the preceding word is *cyngor* (council), as in *cyngor dosbarth* (district council).

`substitute (adj :other:) (adv :else:) ("arall" adj :other:) if (-1 ("rhywbeth") or ("rhywle"));`
Replace the English lemma "other" with "else", and change the part of speech from adjective to adverb, when the lemma is *arall* and the preceding word is *rhywbeth* (something) or *rhywle* (somewhere), as in *rhywbeth arall* (somewhere else).

`substitute (:all:) (:completely:) ("cwbl") if (1 (adj));`
Replace the English lemma "all" with "completely" when the lemma is *cwbl* and the preceding word is an adjective, as in *cwbl sarhaus* (completely insulting).

4.5 Contributing

The results from this Autoglosser2 grammar file are good, but still at an early stage. As it currently stands, the rules are fairly basic – they could be improved or made more abstract using sets. More might also be done to group rules so that rule order has no effect. Further work along these lines would help contribute to a useful set of NLP tools for Welsh, e.g. dependency trees, syntactic analysis, etc. If you would like to improve things, I would be happy to take suggestions on board – fork the repo, make your changes, and then submit a pull request, or send me your revised grammar file directly (kevin@dotmon.com).

[6] The keyword is actually *SUBSTITUTE*, but the most common English construction here is "substitute A for B", which implies ending up with A; the CG-3 authors are using the less common construction "substitute A with B", which implies ending up with B, like the more frequent "replace A with B". This is because *REPLACE* is already being used as a keyword to replace individual tags in a reading.

Chapter 5

Eurfa

5.1 Introduction

Work on Eurfa (Donnelly, 2016) began in 2003, and the first version was released in 2006. It is the largest Welsh dictionary under a free (GPL) license,[1] and it was the first dictionary of a Celtic language to list verbal inflections and mutated forms as headwords. Currently it contains around 10,500 lemmas. Further information, and a web interface, is available at the Eurfa website,[2] and the contents of the dictionary are available at the Git repository[3] – see Appendix A/4. A key aim for Eurfa is that it should be useable in a variety of Welsh NLP situations, so its structure reflects the need for versatility. This chapter summarises the main features of Eurfa as a component of Autoglosser2.

Eurfa was used in the first-ever glosser for Welsh conversational text, the Bangor Autoglosser,[4] developed in 2009-2011 as part of a multilingual spoken corpus project run by the ESRC Centre for Research on Bilingualism at Bangor University. One of these corpora, the Welsh-English *Siarad* corpus, was the first Welsh corpus to be released under a GPL license, and the same license has been applied to the others (a Welsh-Spanish corpus from Patagonia, and an English-Spanish corpus from Miami).

Eurfa has also been used in two other Welsh NLP projects: the Welsh Natural Language Toolkit[5] and the Corpus Cenedlaethol Cymraeg Cyfoes (CorCenCC – the National Corpus of Contemporary Welsh).[6]

Eurfa's function in all these software programs is to provide a listing of possible parts of speech for any given word. This listing can then be pruned to leave (hopefully) one item which applies to the word.

5.2 Structure

The main elements of the `eurfa` database table are exemplified in Table 5.1.

surface	**lemma**	**enlemma**	**pos**	**gender**	**number**	**tense**	**corcencc**	**plural**
coes	coes	leg	n	f	sg		E b u	coesau
dyfroedd	dŵr	water	n	m	pl		E g ll	
mynd	mynd	go	v			infin	Be	
aeth	mynd	go	v		3s	past	B gorff 3 u	
hapus	hapus	happy	adj				Ans cad u	
rhywsut	rhywsut	somehow	adv				Adf	
heb	heb	without	prep				Ar sym	

Table 5.1: *Eurfa structure.*

The structure of Eurfa has changed at various points over the years (as can be seen from its commit history), and it is important to emphasise that this is likely to continue, but the original aim was to create a datasource that was versatile enough to be used in a variety of computerised NLP tasks, but which

[1] `gnu.org/licenses/gpl.html`
[2] `eurfa.org.uk`
[3] `bitbucket.org/donnek/eurfa`
[4] `bangortalk.org.uk/autoglosser.php`
[5] `hypermedia.research.southwales.ac.uk/kos/wnlt`
[6] `corcencc.org`

would be easily editable by non-specialists. This is why the data is maintained in one database table, rather than using multiple tables linked by foreign keys: the layout will be familiar to anyone who has compiled a simple glossary or vocabulary list, and allows the dictionary to be opened in a spreadsheet, if desired, to add new entries or edit existing ones.[7]

The `eurfa` database table contains 17 fields, though not all are used for each entry:

id A numeric identifier for the entry.

surface The form of the word as it might appear in running text – the "surface" form derived from the lemma by the addition of various morphemes.

lemma The underlying (base, stem) form of the word. This is usually the form listed as the headword in a traditional dictionary (e.g. the infinitive for verbs, the singular for nouns),and may be identical to *surface* in many cases. *Lemma* allows related words to be grouped in searches.

enlemma The lemma of the English word equivalent to the Welsh lemma.

pos The part of speech of the word, e.g. adjective, verb, noun, interactional marker, name, punctuation, etc. For the various components in this and the next three fields, see Appendix B.

gender The grammatical gender of the word, e.g. masculine, feminine.

number The grammatical number of the word, e.g. singular, plural. This also includes person for verbs.

tense The tense of the word if it is a verb.

usage Notes on the usage of the word, e.g. spoken form, non-standard form, error, etc.

gramnotes Notes on grammatical aspects of the word, e.g. the form occurs pre-vowel, as a verbal object, before a measurement word, etc.

extended A more detailed definition of the word, such as might occur in a dictionary. This may contain multiple English options, while *lemma* will contain only one (the most common), allowing it to be used in glosses. For instance, *gweddill* has "remainder" in *enlemma* and "remainder, remnant" in *extended*, meaning that we only need to maintain one entry in the dictionary instead of two (one for each meaning), and do not need to add rules in Autoglosser2 ruling out one of them.[8] Additional clarifications may also be added here for people, places or organisations, e.g. the fact that *Estyn* is an education and training inspectorate for Wales.

corcencc The tag for this word as used by CorCenCC. These are generated by programmatically mapping existing Eurfa POS data onto CorCenCC tags – see Appendix C for a full list. In Eurfa, the components of the tag are separated (e.g. *E b u* instead of *Ebu*), and joined together when the tag is printed out - this is to allow easier application of constraint grammar rules if desired.

posplus Additional part of speech information, e.g. whether a name is a toponym or a personal name, the degree of a demonstrative (near, far, etc), the type of punctuation mark, etc. This field is also used to mark things such as incorrect double mutations.

deriv Currently unused – available for future display of etymology.

plural The plural form of nouns where the singular can be predicted from the plural (at least 75% of Welsh singulars are in this category). Plurals which are not predictable are given their own main entry. In Autoglosser2, we de-pluralise words, look up the singular, and then mark the part of speech as plural (see Section 3.5), so the plural column is not used. It is, however, necessary for dictionary use: for any given noun, we search in both the surface field (for non-predictable plurals) and the plural field (for predictable plurals), and if an entry is found in either, we can then display the entry as "(plural of:)" using the lemma field for (for non-predictable plurals) or the surface field (for predictable plurals).

segment The morphological segmentation of the word. This is work in progress, and is only filled in for nouns at the minute. Morpheme boundaries are shown with % (e.g. *iach%awd%wr%i%aeth*, salvation), and word boundaries of combined words with ∼ (e.g. *byd∼enw%og*, world-famous). Morphemes are spelt in their "base" form, even where there are spelling changes in the surface word, e.g. *am%cylch%i%ad* → *amgylchiad* (circumstance), *arwain%ydd* → *arweinydd* (leader). Segment allows derivationally-related words to be grouped in searches.

status Currently unused – available for future development.

[7] Another benefit of this simple word-based approach is that it is possible to plug a wordlist for any language into Autoglosser2 and get some output immediately (though it will not be disambiguated) – this means that it is easy to get started on producing an autoglosser for a new language.

[8] Cases where one English equivalent might be more appropriate than another can be handled by contextual rules in the constraint grammar – see Section 4.4.3.

5.3 Content

Although Eurfa should cover at least 80% of common text, it is inevitable that there will be words in the text that are not in Eurfa. The output pdf flags missing ("unknown") words in red, as does the web output.[9] Adding such words to Eurfa will not only allow them to be glossed, but may also improve the glossing of surrounding words – for instance, if a rule for a word A depends on knowing whether a following word B is a noun or a verb, then word A is likely to remain undisambiguated if the following word B cannot be resolved as either because it is missing from Eurfa.[10]

The approach to content adopted for Autoglosser2 has been to seek to put as much material as possible into the dictionary, so that there is only one source that needs to be edited, rather than multiple files. Punctuation, abbreviations,[11] letters of the alphabet, etc. are therefore all included in Eurfa alongside "normal" words. Eurfa also contains colloquialisms and re-spellings based on actual pronunciation (e.g. *gyn* for *gan*) which were used in the *Siarad* conversations and reflect spoken rather than written Welsh. A few entries also reflect errors or non-standard usage (e.g. *hawsach* (easier) instead of *haws*; *blynyddau* (years) instead of *blynyddoedd*; double mutation) as recorded in *Siarad*.

Note that when used in Autoglosser2 Eurfa has a symbiotic relationship with it. Changing aspects such as tokenisation or cohort generation may require changes in the content (new entries) or structure (new fields) in Eurfa, and these changes may in turn require changes in the rule definitions in the constraint grammar.

5.4 Contributing

If you can improve Eurfa (either by suggesting new features or donating new lexical data), I would be happy to take suggestions on board – fork the repo, make your changes, and then submit a pull request, or send me your suggestions directly (kevin@dotmon.com).

[9] Note that words which are not so marked may also be incorrectly glossed!

[10] The long-term answer to this is of course to add more words to Eurfa, which can be a slow process. It is unfortunate that so little lexical data for Welsh is available under a free (or even open) license, but things are slowly changing.

[11] Abbreviations and acronyms are listed in undotted form (i.e. *BBC* instead of *B.B.C.* or *B. B. C.*). All variants will be glossed, but the gloss will contain the undotted form even if the original word is dotted.

Chapter 6

Customising Autoglosser2

6.1 Introduction

This chapter summarises areas where you can customise or improve the output of Autoglosser2 for your own needs.[1] This is apart from aspects such as improving or adding to the number of constraint grammar rules (Chapter 4), or expanding Eurfa (Chapter 5).

6.2 Web interface

The default web interface has a truncation limit of 300 characters. If you are running Autoglosser2 on a local webserver, you can change this limit by finding this line in *postag_welsh.php*:

```
$instring=strip_tags(substr($instring, 0, 300));
```

and editing it to increase the figure of 300, or commenting it out entirely (by placing two slashes – `//` – at the beginning) to remove any length limit on the text.

The default tiered output of the web interface is 4 words wide. If you want to take advantage of a wider screen, this can be adjusted by finding this line in *txtoutput.php*:

```
$chunks=array_chunk($values, 4, true);
```

and changing "4" to an appropriate number. Bear in mind that you may also need to change the CSS rules governing the layout of the page (`style.css`).

6.3 Import

The segmentation process in `import.php` sets an end-of-sentence marker, and then reverts it in particular cases (e.g. abbreviations, acronyms, instances where the sentence-end comes within quote-marks).

Common abbreviations (e.g. Dr., Ltd., Jan.) are marked so that their final fullstops will not be mistaken as a sentence ending. The list is contained in the *mark_abbrevs()* function in *includes/fns.php*, and can be extended as required.

6.4 Tokenisation

The tokenisation process in `wordify.php` is based on surrounding all non-alphanumeric characters with spaces, and then reverting those spaces in particular cases.[2]

Hyphenated words need special handling, relating especially to the ways in which they may be capitalised. For instance, *cyd-destunoli* (contextualise) would usually be capitalised as *Cyd-destunoli*, but in

[1] Bear in mind that Autoglosser2 is licensed under the GPL, so if you "distribute" the revised code in any way you need to make that revised code available somewhere for download.

[2] My current view is that off-the-shelf tokenisers, many of which require the installation of sizeable libraries (9Mb in one case) do not offer substantially better results than the (8Kb) approach adopted here, but I remain open to persuasion.

headings may also be capitalised as *Cyd-Destunoli*. The tokenisation process joins uncapitalised post-hyphen items to capitalised or uncapitalised pre-hyphen components, so in this case *destunoli* will be joined to the component *Cyd*, and the whole hyphenated word will then be looked up in Eurfa as *cyd-destunoli*. This approach allows compounds such as *Llafur-Plaid* (Labour-Plaid [Cymru]) or *Abertawe-Caerdydd* (Swansea-Cardiff) to be left unjoined, with each side of the compound being looked up individually. However, this also means that *Cyd-Destunoli* will be treated as a compound, and the lookup will fail or be incorrect. To handle this, a specific subset of components (e.g. *cyd-*, *di-*, *ôl-*) is listed in the *hyphcomp()* function in *includes/fns.php*, and can be extended as required.

6.5 Cohort generation

Lookup routines for different languages can be added to `cohorts.php`, either replacing `lookups/cym_lookup.php`, or augmenting it with lookups for different languages. The Bangor Autoglosser used the latter approach, with the dictionary to be looked up depending on the language tag assigned to the word by the transcriber of the conversation, but it would also be possible simply to look up the word successively in all the dictionaries. This is done simplistically in Autoglosser2, where an English wordlist (*saesneg*)[3] is looked up as part of `lookups/cym_lookup.php` to identify possible English words, but unlike the fuller lookup in the Bangor Autoglosser, no parts of speech are given for those words. This behaviour can be adjusted by augmenting the English dictionary to include parts of speech (for instance, by swapping in the one from the Bangor Autoglosser).

What each lookup file does can also be customised. For instance, the URL identification in the default Welsh lookup is simplistic, and only covers the most basic types of URL.

The layout of the cohorts file is also variable within limits – the constraint grammar parser offers some flexibility as regards the format of the file it will accept. This means that data (eg location data) can be written to that file for use outside constraint grammar itself.

6.6 Gloss generation

The data in the `_cgfinished` table can be formatted in a variety of ways, so it is possible to generate glosses or tags in a particular format to suit your application.

If you need a completely different tagset, that can be created by adding another column to Eurfa and adding the tags programmatically to each word (as was done for the CorCenCC project). They can then be used in the pipeline at the cohort generation stage.

6.7 Pdf output

The colours for the glosses or tags can be edited in *pdfoutput.php*:

```
$gcolour="Blue";
$tcolour="Green";
```

The output *.tex* file includes Uwe Kern's *xcolor* package,[4] so a wide range of colours is available, as listed in Section 4 of the *xcolor* manual.[5]

John Frampton's ExPex package[6] is used for the alignment of the glosses or tags with the original text. This package offers many alternatives, and since the glossing data is contained in a database table it can be output programmatically in various different formats to leverage those alternatives.

[3] Based on Alan Beale's 12dicts 5d + 2a list – `wordlist.aspell.net/12dicts-readme/#internat`.
[4] `ctan.org/pkg/xcolor`
[5] `mirrors.ctan.org/macros/latex/contrib/xcolor/xcolor.pdf`
[6] `ctan.org/pkg/expex`

LaTeX in general offers a multitude of possibilities for elegant typesetting, so there are many ways in which the glossed data can be presented to readers in an attractive format, or used in academic papers or books – for a demonstration of its versatility outside the field of part of speech tagging, see Donnelly (2017, Chapter 8).

6.8 Txt output

The web and text output can be adjusted to suit your own needs, and the text output can be ingested directly by other programs, so that (for instance) syntactical structure or semantic domain can be further investigated.[7]

XML output is also possible, but is not offered by default, largely because there is no consensus on what a "canonical" XML file for NLP work should look like.[8]

6.9 Contributing

If you would like to improve Autoglosser2, I would be happy to take suggestions on board – fork the repo, make your changes, and then submit a pull request, or send me your code directly (kevin@dotmon.com). The coding style is pretty obvious: blank lines between "sections" of code, plenty of comments, and NO brace at the end of lines – put them on a new line. I also prefer procedural style with functions rather than object-oriented, but I'm happy to be convinced otherwise.

[7] Of course, it is also possible to manipulate the database tables directly to create new tables for particular purposes – see, for instance, Broersma et al. (2018) and Deuchar et al. (2016).

[8] In my view, there are other reasons to avoid using XML in NLP work in any case.

References

Bick, E. (2009). Basic Constraint Grammar tutorial for CG-3 (Vislcg3). beta.visl.sdu.dk/cg3_howto.pdf.

Bick, E. and T. Didriksen (2015). CG--3: Beyond classical Constraint Grammar. In *Proceedings of NODALIDA 2015, Vilnius, Lithuania*. aclweb.org/anthology/W15-1807.

Broersma, M., D. Carter, and K. Donnelly (2018). Triggered codeswitching: Lexical processing and conversational dynamics. *Bilingualism: Language and Cognition*. Forthcoming.

Caliskan-Islam, A., J. J. Bryson, and A. Narayanan (2017). Semantics derived automatically from language corpora necessarily contain human biases. *Science 356*, 183–186.

Carter, D., M. Broersma, and K. Donnelly (2016). Applying computing innovations to bilingual corpus analysis. In A. Alba de la Fuente, E. Valenzuela, and C. Martínez-Sanz (Eds.), *Language Acquisition Beyond Parameters: Studies in Honour of Juana M. Liceras*, Number 51 in Studies in Bilingualism, pp. 281–301. Amsterdam: John Benjamins.

Carter, D., M. Broersma, K. Donnelly, and A. Konopka (2017). Presenting the Bangor Autoglosser and the Bangor Automated Clause Splitter. *Digital Scholarship in the Humanities*.

Comrie, B., M. Haspelmath, and B. Bickel (2008). Leipzig glossing rules: conventions for interlinear morpheme-by-morpheme glosses. eva.mpg.de/lingua/resources/glossing-rules.php.

Deuchar, M., K. Donnelly, and C. Piercy (2016). Mae pobl monolingual yn minority: Factors favouring the production of code-switching by Welsh-English speakers. In M. Durham and J. Morris (Eds.), *Sociolinguistics in Wales*, pp. 209–239. London: Palgrave Macmillan.

Deuchar, M., P. Webb-Davies, and K. Donnelly (2018). *Building and Using the Siarad Corpus: Bilingual conversations in Welsh and English*. Number 81 in Studies in Corpus Linguistics. Amsterdam: John Benjamins. Forthcoming.

Didriksen, T. (2017). Constraint Grammar Manual. beta.visl.sdu.dk/cg3/chunked.

Donnelly, K. (2010). Getting started with Constraint Grammar. kevindonnelly.org.uk/resources/tutorial.pdf.

Donnelly, K. (2013). Kynulliad3: a corpus of 350,000 aligned Welsh-English sentences from the Third Assembly (2007-2011) of the National Assembly for Wales. cymraeg.org.uk/kynulliad3.

Donnelly, K. (2016). Eurfa: a free (GPL) dictionary for Welsh, v3. eurfa.org.uk.

Donnelly, K. (2017). Writing and transliterating Swahili in Arabic script with *Andika!* kevindonnelly.org.uk/swahili/manual_draft.pdf.

Donnelly, K., S. Cooper, and M. Deuchar (2011). Glossing CHAT files using the Bangor Autoglosser. *8th International Symposium for Bilingualism, Oslo, Norway*. kevindonnelly.org.uk/resources/words/Donnelly2011_ISB8.pdf.

Donnelly, K. and M. Deuchar (2011a). The Bangor Autoglosser: a multilingual tagger for conversational text. In *Proceedings of the Fourth International Conference on Internet Technologies and Applications (ITA11)*, Wrexham, Wales. kevindonnelly.org.uk/resources/words/Donnelly2011_Autoglosser.pdf.

Donnelly, K. and M. Deuchar (2011b). Using constraint grammar in the Bangor Autoglosser to disambiguate multilingual spoken text. In *Constraint Grammar Applications: Proceedings of the NODALIDA 2011 Workshop, Riga, Latvia*, NEALT Proceedings Series, Tartu, pp. 17–25. hdl.handle.net/10062/19298.

Karlsson, F. (1990). Constraint grammar as a framework for parsing running text. In *COLING-90: Proceedings of the 13th Conference on Computational Linguistics*, Somerset, NJ, pp. 168–173. Association for Computational Linguistics. anthology.aclweb.org/C/C90/C90-3030.pdf.

Karlsson, F., A. Voutilainen, J. Heikkilä, and A. Anttila (1995). *Constraint Grammar: a Language-Independent System for Parsing Unrestricted Text*. Berlin: Mouton de Gruyter.

Appendix A

Installing Autoglosser2

A/1 Introduction

This appendix explains how to install from scratch the software that Autoglosser2 needs. The installation has been tested on Ubuntu Linux 14.04 and 16.04,[1] and it should run on other platforms in a virtual machine.[2] If you are having difficulties getting something installed, please email me at kevin@dotmon.com – although I have tried the instructions below on a "clean" machine, I may nevertheless have missed something, or got the sequence wrong.

Most of the install is carried out by typing commands directly into a terminal or console. This is because this method is much faster and more succinct than explaining how to point-and-click through various dialogue boxes.

A/2 Conventions

Unless otherwise indicated, lines in
```
monospaced font
```
are commands to be typed in.

Unless otherwise indicated, all commands should be activated by pressing *Return* at the end of the command.

The symbol ↪ at the beginning of a line means that it is a continuation of the previous command, and therefore *Return* should only be pressed after the end of this line.

Keys separated by + should be pressed simultaneously. Thus *Ctrl*+*X* means "press the Ctrl key at the same time as the X key".

When a command starts with *sudo*, you will be asked to type in your superuser (administrative) password, which you should have been asked to set up when you first installed Ubuntu, before the command is allowed to proceed. Note that you will get no feedback from the password entry (the line will stay blank) until you press *Return*.

If at any point the system suggests adding other packages (called *dependencies*) based on the ones you are installing, accept those suggestions by pressing *Y* or typing *yes*.

Unless otherwise indicated, it is assumed that all commands are run from the suggested base directory for Autoglosser2, */var/www/autoglosser* – see Section A/3.

A/3 Download Autoglosser2

The main reason you're here! First, install Git, which keeps track of changes made to files:
```
sudo apt-get install git
```
and then move to your home directory (USER here stands for the username you set up when you installed

[1] `ubuntu.com`
[2] e.g. `virtualbox.org`

Ubuntu; replace it with your actual username),[3] and download Autoglosser2:

```
cd /home/USER
git clone https://bitbucket.org/donnek/autoglosser2.git
```

After a minute or two, Autoglosser2 will be downloaded into an *autoglosser2* folder in */home/USER*. In the future, if you want to update Autoglosser2, you can open a terminal in the *autoglosser2* folder and type:

```
git pull
```

Git will automatically update those parts of Autoglosser2 which have been changed.

To allow for the web interface (for configuration, see Appendix A/8), move the *autoglosser2* directory to */var/www*, the default location for storing webpages on Ubuntu:

```
sudo mv autoglosser2 /var/www/
```

Note the final slash. Give yourself (USER stands for your own username) ownership of the */var/www* directory;

```
sudo chown -R USER.USER /var/www/
```

and then set up a link from your */home/USER* directory to the */var/www/* directory:

```
ln -s /var/www/ web
```

Now when you go to */home/USER/web* in a file manager, it will take you to */var/www/*. Move into the *autoglosser2* directory for the rest of the installation:

```
cd web/autoglosser2
```

A/4 Download Eurfa

Eurfa provides a dictionary giving part of speech information about Welsh words in your text:

```
git clone https://bitbucket.org/donnek/eurfa.git
```

These files will be installed later as database tables (see Appendix A/11).

A/5 Install fonts

These fonts are used in the pdf output:

```
sudo apt-get install fonts-sil-charis fonts-liberation fonts-linuxlibertine
    ↪ fonts-dejavu
```

A/6 Install Constraint Grammar 3

CG-3 will apply constraint grammar rules to a file of word cohorts. First, install wget:

```
sudo apt-get install wget
```

and add the repository signing key to your keyring:

```
sudo wget http://apertium.projectjj.com/apt/apertium-packaging.public.gpg
    ↪ -O /etc/apt/trusted.gpg.d/apertium.gpg
```

Then add the relevant Ubuntu CG-3 repository (replace *DISTRO* with your Ubuntu version, e.g. *trusty*, *wily*):

```
echo "deb http://apertium.projectjj.com/apt/nightly DISTRO main"
    ↪ sudo tee /etc/apt/sources.list.d/apertium-nightly.list
```

and update the repository list:

```
sudo apt-get update
```

so that you can install CG-3 and its integrated development environment (IDE):

```
sudo apt-get install cg3 cg3ide
```

[3] The terminal prompt will tell you what your username is – it is of the form *user@computer*.

A/7 Install Apache2

Apache2 is a webserver which displays the Autoglosser2 web interface:

```
sudo apt-get install apache2 apache2-utils phppgadmin
```

Start the webserver:[4]

```
sudo service apache2 start
```

and then open a web browser (preferably Firefox) and enter:

```
http://localhost
```

into the address bar. A page will open, telling you that Apache2 is installed and working.

A/8 Configure the web interface

Tell Apache2 where to find the Autoglosser2 webpages by opening a new configuration file:

```
sudo nano /etc/apache2/sites-available/autoglosser2.conf
```

Nano is a lightweight text editor: use the arrow keys on the keyboard to move around, and the *Home* and *End* keys to move to the beginning or end of a line. Type in the following lines:

```
<VirtualHost *:80>
ServerName autoglosser2
DocumentRoot /var/www/autoglosser2/
</VirtualHost>
```

Save the file: press *Ctrl*+*X*, then press *Y* to confirm you want to save the modifications, and press *Return* to close the file. Activate the configuration:

`sudo a2ensite autoglosser2` and restart the webserver:

```
sudo service apache2 restart
```

Tell your web browser that the new website is on your machine (so it will not try to look for it on the web) by opening the *hosts* configuration file:

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

After the line:

```
127.0.0.1 localhost
```

add the following line:

```
127.0.0.1 autoglosser2
```

Then save and exit the configuration file (*Ctrl*+*X* → *Y* → *Return*), and type:

```
http://autoglosser2
```

into the address bar of your browser. You will get the Autoglosser2 web interface from */var/www/autoglosser2*.

A/9 Install PHP

PHP is the scripting language used by Autoglosser2:

```
sudo apt-get install php5 php5-cli
```

Once installed, open the PHP command-line configuration file so that you can change some settings:

```
sudo nano /etc/php5/cli/php.ini
```

Press *Ctrl*+*W*, then type

```
max_execution_time
```

into the searchline and press *Return*. Change the line to read:

```
max_execution_time = 300
```

Press *Ctrl*+*W* again, and type

```
error_reporting
```

and press *Return*. Change that line to:

```
error_reporting = E_ALL & ~E_NOTICE & ~E_DEPRECATED
```

[4] If you want to get rid of the (harmless) message "Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message", issue the following commands:

```
echo "ServerName localhost" | sudo tee /etc/apache2/conf-available/servername.conf
sudo a2enconf servername
sudo service apache2 restart
```

Scroll down (using the mouse or *down* arrow) to the *display_errors* line a bit lower down. Change it to read:

```
display_errors = On
```

Below that there is a *log_errors* line. Change it to read:

```
log_errors = Off
```

Save and exit the configuration file (*Ctrl*+*X* → *Y* → *Return*)

PHP is an easy computer language to get to grips with, gives immediate results, and works on both the browser and the console, but describing it lies outside the scope of this manual. There are many tutorial sources on the internet.

A/10 Install PostgreSQL

PostgreSQL is the database that will handle the storage your text data:

```
sudo apt-get install postgresql postgresql-client postgresql-common
 ↪ postgresql-contrib php5-pgsql
```

On Ubuntu, PostgreSQL uses peer authentication by default. Creating a database user with the same name as your system (Ubuntu) user will therefore mean that you can log in to the database without entering a password. To set yourself up as the database user, become the *postgres* master user:

```
sudo su - postgres
```

(note the space on either side of the dash). Enter your superuser (administrator) password. The prompt will change to *postgres@computer*. Create a new database user with the same name as your username (USER below):

```
createuser -P -s -e USER
```

Enter a password – note that you will get no feedback (the line will stay blank). Press *Return*, and enter the password again. Press *Return* and you should get a message beginning *CREATE ROLE*, meaning that the new user has been created. Go back to being your normal user:

```
exit
```

The Structured Query Language (SQL) used by relational databases like PostgreSQL is very powerful and flexible, but describing it lies outside the scope of this manual. There are many tutorial sources on the internet.

A/11 Configure the database connection

Configure Autoglosser2 to use your new database account by opening the configuration file:

```
nano autoglosser2/config.php
```

and changing:

```
user=kevin password=kevindbs
```

to read:

```
user=USER password=yourpassword
```

(remember to replace USER with your username). You need to edit two lines (one for the *autoglosser2* database and one for the *eurfa* database), and also edit the password (*PGPASSWORD*=) and username (*-U*) on the third line (the one for the *psql* shell). Exit the configuration file (*Ctrl*+*X* → *Y* → *Return*), and then move the configuration data outside the web directory:

```
sudo mv autoglosser2/ /opt/
```

entering your superuser password when asked. The database connection file is now at */opt/autglosser2*, which is where the other scripts in Autoglosser2 expect to find it.

Create the *autoglosser2* database to hold the data you want to gloss:

```
createdb autoglosser2
```

and import the English wordlist:

```
psql -d autoglosser2 < dbs/saesneg.sql
```

Create the *eurfa* database to hold the Welsh dictionary:

```
createdb eurfa
```

Import the Eurfa data (downloaded in Appendix A/4) into the database:

```
psql -d eurfa < eurfa/eurfa.sql
```

A/12 Install phpPgAdmin

phpPgAdmin is a browser interface to PostgreSQL – it will make it easier to inspect your database tables:

```
sudo apt-get install phppgadmin
```

Activate the phpPgAdmin configuration file:

```
sudo cp /etc/apache2/conf.d/phppgadmin /etc/apache2/conf-enabled/phppgadmin.conf
```

and then restart the webserver:

```
sudo service apache2 restart
```

In a web browser, enter:

```
http://localhost/phppgadmin
```

into the address bar. You should see the phpPgAdmin homepage. On the left side there is a list of servers. Click on *PostgreSQL* and you should get a login form. Fill in the username and password for PostgreSQL (which you created in Appendix A/10) and click *Login*. In the left-hand panel you should get a list of your current databases – there should be three: *autoglosser2*, *eurfa*, and *postgres* (the system database, which you will not be using).

Click the + beside *eurfa* in the left-hand panel. It should open to show *Schemas, public, Tables* etc. Click on *Tables*. The right-hand panel should now show you the (only) table – *eurfa* – inside the *eurfa* database. Click on *eurfa* and you will see the data fields in that table. To see the contents of the table you can click on the *Browse* button. To export the table to csv (which can be opened in a spreadsheet), click *Export* in the top button bar. The *SQL* link at the top right of the phpPgAdmin window can be used to make SQL queries to the database.

You are likely to be using the *autoglosser2* database rather than the *eurfa* database, but the same principles apply there when you want to look at the tables created by the glossing process.

The default session time for phpPgAdmin is set to 24 minutes (1440 seconds), meaning that if you do not use phpPgAdmin for 24 minutes, it will ask you to log in again before you can continue using it. You can change this by opening the PHP web configuration file:

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

Press *Ctrl* + *W*, then type:

```
session.gc_maxlifetime
```

and press *Return*. Change the line to read:

```
session.gc_maxlifetime = 144000
```

This will allow you 40 hours before logging you out, which should be sufficient.

A/13 Install SQL Workbench/J

SQL Workbench/J is another interface to PostgreSQL – its main benefit is that a result set can be directly edited to update the data. It uses Java, and you can begin installing the Oracle version of Java by adding Andrei Alin's repository:

```
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:webupd8team/java
```

and updating your software package list:

```
sudo apt-get update
```

Then install the Java installer:

```
sudo apt-get install oracle-java8-installer
```

which installs a script that downloads and installs Java 8 from the Oracle website. You will be asked to accept the Oracle license before the installation begins. Once installed, runningl:

```
java -version
```

should return some text telling you that the Java version is 1.8.0. Now install software to allow SQL

Workbench to connect to the PostgreSQL database:

```
sudo apt-get install libpostgresql-jdbc-java
```

Now you can install SQL Workbench/J itself. Create a subdirectory for it:

```
mkdir sqlworkbench
```

and then go to the website *sql-workbench.net*, click on the link for *Build 123* (or whatever the current stable version is), and download the generic package. Save it in the *autoglosser2* directory, and then unzip the download into the new directory:

```
unzip -q Workbench-Build123.zip -d sqlworkbench
```

Make the launch script executable:

```
chmod +x sqlworkbench/sqlworkbench.sh
```

and launch SQL Workbench/J:[5]

```
sqlworkbench/sqlworkbench.sh
```

A/14 Configure SQL Workbench/J

The first thing you will see is a *Select Connection Profile* box, where you need to add the details of the Autoglosser2 database. Change *New profile* to read *autoglosser2*. Click the drop-down arrow on the *Driver* line and select *PostgreSQL*. Click *Yes* when you're asked whether you want to edit the driver definition.

On the *Manage Drivers* popup, click on the red *postgresql* entry already there and then click X to delete it. Click on the folder icon and navigate to */usr/share/java/postgresql-jdbc4-9.2.jar* (which you just installed – Appendix A/13). Click *Open*, and then *OK*. Check that the *URL* line reads:

```
jdbc:postgresql://localhost:5432/andika
```

If not, edit it to make it so. Enter your PostgreSQL username and password (Section A/10), and then click *OK*. You will get a connecting message.

If you like, you can also set up a connection to the Eurfa database, which will allow you to add new entries and edit existing ones. Press Alt + C to bring up the *Select Connection Profile* box again. Click the first button in the left-hand panel (*Create a new connection profile*), and change *New profile* to read *eurfa*. Then proceed as for the *autoglosser2* database.

Once both connection profiles are set up, select *File* → *Save Profiles* to save them.

A/15 Install LaTeX

LaTeX is an unbelievably versatile typesetting system that provides attractive pdf output in Autoglosser2:

```
sudo apt-get install texlive texlive-base texlive-latex-base texlive-latex-recommended
  ↪ texlive-latex-extra texlive-xetex texlive-generic-extra texlive-humanities
  ↪ texlive-publishers texlive-extra-utils texlive-pstricks texlive-bibtex-extra
  ↪ kile kbibtex biber
```

Note that these packages will take perhaps 20 minutes to download and install.

[5] You can make a desktop shortcut or menu entry to make launching SQL Workbench easier.

Appendix B

Autoglosser2 gloss components

Component	Description
0	impersonal
1P	1st person plural
1S	1st person singular
2P	2nd person plural
2S	2nd person singular
3P	3rd person plural
3S	3rd person singular
ABB	abbreviation
ACR	acronym
ADJ	adjective
ADV	adverb
AFF	affirmative
am	aspirate mutation
ANCL	closing angle bracket
ANOP	opening angle bracket
AR	Arabic
AUG	augmentative
BRCL	closing brace (curly bracket)
BROP	opening brace (curly bracket)
CO	comma
COL	colon
COMP	comparative
COND	conditional
CONJ	conjunction
CYR	Cyrillic glyphs
DEC	decimal
DEF	definite
DEM	demonstrative
DET	determiner
DIM	diminutive
E	exclamation
ELL	ellipsis
EM	emdash
EMPH	emphatic
EN	endash
ENG	English
EXCL	exclamation mark
F	feminine
FAR	far (demonstrative)
FOCUS	item with focus
FS	fullstop
FUT	future
h	pre-vocalic h- after 3S.F, 1P and 3P possessives
HAN	Han glyphs
HY	hyphen
HYP	hypothetical
IM	interactional marker

Component	**Description**
IMPER	imperative
IMPERF	imperfect
INFIN	infinitive
INT	interrogative
LET	letter
M	masculine
MF	masculine or feminine
MISC	miscellaneous (unspecified)
N	noun
NAME	name
NEAR	near (demonstrative)
NEG	negative
nm	nasal mutation
NUM	numeral
OBJ	object
ORD	ordinal
PACL	closing parenthesis
PAOP	opening parenthesis
PAST	past
PL	plural
PLUPERF	pluperfect
POSS	possessive
PREP	preposition
PREQ	pre-qualifier
PRES	present
PRON	pronoun
PRT	particle
PUNC	punctuation
QDB	double quote
QSG	single quote
QST	question mark
QUAN	quantifier
REFL	reflexive
REL	relative
ROM	Roman
SCOL	semi-colon
SG	singular
sm	soft mutation
SP	singular or plural
SQCL	closing square bracket
SQOP	opening square bracket
SUBJ	subjunctive
SUP	superlative
TAG	tag question
TOP	toponym (placename)
UNK	unknown
URL	universal resource locator (web address)
V	verb
YR	year

Appendix C

CorCenCC tags

Tag	Disgrifiad	Description	Enghraifft/example
Egu	Enw, gwrywaidd unigol	Noun, masculine singular	ci
Ebu	Enw, benywaidd unigol	Noun, feminine singular	cath
Egll	Enw, gwrywaidd lluosog	Noun, masculine plural	cŵn
Ebll	Enw, benywaidd lluosog	Noun, feminine plural	cathod
Egbu	Enw, gwrywaidd/benywaidd unigol	Noun, masculine/feminine singular	meddalwedd
Egbll	Enw, gwrywaidd/benywaidd lluosog	Noun, masculine/feminine plural	arferion
Epg	Enw, priod, gwrywaidd	Noun, proper, masculine	Dafydd
Epb	Enw, priod, benywaidd	Noun, proper, feminine	Cymru
YFB	Y fannod benodol	Definite article	y, yr, ‘r
Arsym	Arddodiad, syml	Preposition, uninflected	ar
Ar1u	Arddodiad rhediadol, person 1af unigol	Inflected preposition, 1st person singular	arnaf
Ar2u	Arddodiad rhediadol, 2il person unigol	Inflected preposition, 2nd person singular	arnat
Ar3gu	Arddodiad rhediadol, 3ydd person gwrywaidd unigol	Inflected preposition, 3rd person masculine singular	arno
Ar3bu	Arddodiad rhediadol, 3ydd person benywaidd unigol	Inflected preposition, 3rd person feminine singular	arni
Ar1ll	Arddodiad rhediadol, person 1af lluosog	Inflected preposition, 1st person plural	arnom
Ar2ll	Arddodiad rhediadol, 2il person lluosog	Inflected preposition, 2nd person plural	arnoch
Ar3ll	Arddodiad rhediadol, 3ydd person lluosog	Inflected preposition, 3rd person plural	arnynt
Cyscyd	Cysylltair cydradd	Coordinating conjunction	a, ac
Cysis	Cysylltair isradd	Subordinating conjunction	ers, gan fod
Rhifol	Rhifair, rhifolyn (0-10)	Numeral, cardinal (0-10)	saith, dau, dwy
Rhifold	Rhifair, rhifolyn, degol (11+)	Numeral, cardinal, decimal (11+)	un deg un
Rhifolt	Rhifair, rhifolyn, traddodiadol (11+)	Numeral, cardinal, traditional (11+)	ugain, dau ar hugain, dwy ar hugain
Rhitref	Rhifair, trefnolyn (0-10)	Numeral, ordinal (0-10)	seithfed, pedwerydd, pedwaredd
Rhitrefd	Rhifair, trefnolyn, degol (11+)	Numeral, ordinal, decimal (11+)	un deg wythfed
Rhitreft	Rhifair, trefnolyn, traddodiadol (11+)	Numeral, ordinal, traditional (11+)	deunawfed
Anscadu	Ansoddair cadarnhaol, unigol	Adjective, positive, singular	byr
Anscadbu	Ansoddair cadarnhaol benywaidd unigol	Adjective, positive, feminine singular	ber
Anscadll	Ansoddair cadarnhaol, lluosog	Adjective, positive, plural	byrion

Tag	Disgrifiad	Description	Enghraifft/example
Anscyf	Ansoddair, cyfartal	Adjective, equative	byrred
Anscym	Ansoddair, cymharol	Adjective, comparative	byrrach
Anseith	Ansoddair, eithaf	Adjective, superlative	byrraf
Adf	Adferf	Adverb	adref, felly
Be	Berf enw	Verb noun	ariannu
Bpres1u	Berf, presennol, person 1af unigol	Verb, present, 1st person singular	ariannaf
Bpres2u	Berf, presennol, 2il person unigol	Verb, present, 2nd person singular	arienni
Bpres3u	Berf, presennol, 3ydd person unigol	Verb, present, 3rd person singular	arianna
Bpres1ll	Berf, presennol, person 1af lluosog	Verb, present, 1st person plural	ariannwn
Bpres2ll	Berf, presennol, 2il person lluosog	Verb, present, 2nd person plural	ariennwch
Bpres3ll	Berf, presennol, 3ydd person lluosog	Verb, present, 3rd person plural	ariannant
Bpresamhers	Berf, presennol, amhersonol	Verb, present, impersonal	ariennir
Bpres3perth	Berf, presennol, 3ydd person perthynnol	Verb, present, 3rd person singular relative	sy, sydd
Bdyf1u	Berf, dyfodol, person 1af unigol	Verb, future, 1st person singular	byddaf
Bdyf2u	Berf, dyfodol, 2il person unigol	Verb, future, 2nd person singular	byddi
Bdyf3u	Berf, dyfodol, 3ydd person unigol	Verb, future, 3rd person singular	bydd
Bdyf1ll	Berf, dyfodol, person 1af lluosog	Verb, future, 1st person plural	byddwn
Bdyf2ll	Berf, dyfodol, 2il person lluosog	Verb, future, 2nd person plural	byddwch
Bdyf3ll	Berf, dyfodol, 3ydd person lluosog	Verb, future, 3rd person plural	byddant
Bdyfamhers	Berf, dyfodol amhersonol	Verb, future, impersonal	byddir
Bgorb1u	Berf, gorberffaith, person 1af unigol	Verb, pluperfect, 1st person singular	arianaswn
Bgorb2u	Berf, gorberffaith, 2il person unigol	Verb, pluperfect, 2nd person singular	arianasit
Bgorb3u	Berf, gorberffaith, 3ydd person unigol	Verb, pluperfect, 3rd person singular	arianasai
Bgorb1ll	Berf, gorberffaith, person 1af lluosog	Verb, pluperfect, 1st person plural	arianasem
Bgorb2ll	Berf, gorberffaith, 2il person lluosog	Verb, pluperfect, 2nd person plural	arianasech
Bgorb3ll	Berf, gorberffaith, 3ydd person lluosog	Verb, pluperfect, 3rd person plural	arianasent
Bgorbamhers	Berf, gorberffaith, amhersonol	Verb, pluperfect, impersonal	arianasid
Bamhen1u	Berf, amhenodol, person 1af unigol	Verb, imperfect, 1st person singular	ariannwn
Bamhen2u	Berf, amhenodol, 2il person unigol	Verb, imperfect, 2nd person singular	ariannet
Bamhen3u	Berf, amhenodol, 3ydd person unigol	Verb, imperfect, 3rd person singular	ariannai
Bamhen1ll	Berf, amhenodol, person 1af lluosog	Verb, imperfect, 1st person plural	ariannem
Bamhen2ll	Berf, amhenodol, 2il person lluosog	Verb, imperfect, 2nd person plural	ariannech

Tag	Disgrifiad	Description	Enghraifft/example
Bamhen3ll	Berf, amhenodol, 3ydd person lluosog	Verb, imperfect, 3rd person plural	ariannent
Bamhenamhers	Berf, amhenodol, amhersonol	Verb, imperfect, impersonal	ariennid
Bgorff1u	Berf, gorffennol, person 1af unigol	Verb, past, 1st person singular	ariennais
Bgorff2u	Berf, gorffennol, 2il person unigol	Verb, past, 2nd person singular	ariennaist
Bgorff3u	Berf, gorffennol, 3ydd person unigol	Verb, past, 3rd person singular	ariannodd
Bgorff1ll	Berf, gorffennol, person 1af lluosog	Verb, past, 1st person plural	arianasom
Bgorff2ll	Berf, gorffennol, 2il person lluosog	Verb, past, 2nd person plural	arianasoch
Bgorff3ll	Berf, gorffennol, 3ydd person lluosog	Verb, past, 3rd person plural	arianasant
Bgorffamhers	Berf, gorffennol, amhersonol	Verb, past, impersonal	ariannwyd
Bgorffsef	Berf, gorffennol, sefydlog	Verb, past, invariant	ddaru
Bgorch2u	Berf, gorchmynnol, 2il person unigol	Verb, imperative, 2nd person singular	arianna
Bgorch3u	Berf, gorchmynnol, 3ydd person unigol	Verb, imperative, 3rd person singular	arianned
Bgorch1ll	Berf, gorchmynnol, person 1af lluosog	Verb, imperative, 1st person plural	ariannwn
Bgorch2ll	Berf, gorchmynnol, 2il person lluosog	Verb, imperative, 2nd person plural	ariennwch
Bgorch3ll	Berf, gorchmynnol, 3ydd person lluosog	Verb, imperative, 3rd person plural	ariannent
Bgorchamhers	Berf, gorchmynnol, amhersonol	Verb, imperative, impersonal	arianner
Bdibdyf1u	Berf, dibynnol dyfodol, person 1af unigol	Verb, subjunctive, 1st person singular	ariannwyf
Bdibdyf2u	Berf, dibynnol dyfodol, 2il person unigol	Verb, subjunctive, 2nd person singular	ariennych
Bdibdyf3u	Berf, dibynnol dyfodol, 3ydd person unigol	Verb, subjunctive, 3rd person singular	arianno
Bdibdyf1ll	Berf, dibynnol dyfodol, person 1af lluosog	Verb, subjunctive, 1st person plural	ariannom
Bdibdyf2ll	Berf, dibynnol dyfodol, 2il person lluosog	Verb, subjunctive, 2nd person plural	ariannoch
Bdibdyf3ll	Berf, dibynnol dyfodol, 3ydd person lluosog	Verb, subjunctive, 3rd person plural	ariannont
Bdibdyfamhers	Berf, dibynnol dyfodol, amhersonol	Verb, subjunctive, impersonal	arianner
Bamod1u	Berf, amodol, person 1af unigol	Verb, conditional, 1st person singular	byddwn
Bamod2u	Berf, amodol, 2il person unigol	Verb, conditional, 2nd person singular	byddit, byddet
Bamod3u	Berf, amodol, 3ydd person unigol	Verb, conditional, 3rd person singular	byddai
Bamod1ll	Berf, amodol, person 1af lluosog	Verb, conditional, 1st person plural	byddem
Bamod2ll	Berf, amodol, 2il person lluosog	Verb, conditional, 2nd person plural	byddech
Bamod3ll	Berf, amodol, 3ydd person lluosog	Verb, conditional, 3rd person plural	byddent, bydden

Tag	Disgrifiad	Description	Enghraifft/example
Bamodamhers	Berf, amodol, amhersonol	Verb, conditional, impersonal	byddid
Rhapers1u	Rhagenw, personol person 1af unigol	Pronoun, personal, 1st person singular	mi, myfi
Rhapers2u	Rhagenw, personol, 2il person unigol	Pronoun, personal, 2nd person singular	ti, tydi
Rhapers3gu	Rhagenw, personol, 3ydd person gwrywaidd unigol	Pronoun, personal, 3rd person masculine singular	ef, efe
Rhapers3bu	Rhagenw, personol, 3ydd person benywaidd unigol	Pronoun, personal, 3rd person feminine singular	hi, hyhi
Rhapers1ll	Rhagenw, personol, person 1af lluosog	Pronoun, personal, 1st person plural	ni, nyni
Rhapers2ll	Rhagenw, personol, 2il person lluosog	Pronoun, personal, 2nd person plural	chi, chwychwi
Rhapers3ll	Rhagenw, personol, 3ydd person lluosog	Pronoun, personal, 3rd person plural	hwy, hwynt–hwy
Rhadib1u	Rhagenw, dibynnol, person 1af unigol	Pronoun, dependent, 1st person singular	fy, ‘m
Rhadib2u	Rhagenw, dibynnol, 2il person unigol	Pronoun, dependent, 2nd person singular	dy, ‘th
Rhadib3gu	Rhagenw, dibynnol, 3ydd person gwrywaidd unigol	Pronoun, dependent, 3rd person masculine singular	ei, ‘i
Rhadib3bu	Rhagenw, dibynnol, 3ydd person benywaidd unigol	Pronoun, dependent, 3rd person feminine singular	ei, ‘i
Rhadib1ll	Rhagenw, dibynnol, person 1af lluosog	Pronoun, dependent, 1st person plural	ein, ‘n
Rhadib2ll	Rhagenw, dibynnol, 2il person lluosog	Pronoun, dependent, 2nd person plural	eich, ‘ch
Rhadib3ll	Rhagenw, dibynnol, 3ydd person lluosog	Pronoun, dependent, 3rd person plural	eu, ‘u
Rhamedd1u	Rhagenw, meddiannol, person 1af unigol	Pronoun, possessive, 1st person singular	eiddof
Rhamedd2u	Rhagenw, meddiannol, 2il person unigol	Pronoun, possessive, 2nd person singular	eiddot
Rhamedd3gu	Rhagenw, meddiannol, 3ydd person gwrywaidd unigol	Pronoun, possessive, 3rd person masculine singular	eiddo
Rhamedd3bu	Rhagenw, meddiannol, 3ydd person benywaidd unigol	Pronoun, possessive, 3rd person feminine singular	eiddi
Rhamedd1ll	Rhagenw, meddiannol, person 1af lluosog	Pronoun, possessive, 1st person plural	eiddom
Rhamedd2ll	Rhagenw, meddiannol, 2il person lluosog	Pronoun, possessive, 2nd person plural	eiddoch
Rhamedd3ll	Rhagenw, meddiannol, 3ydd person lluosog	Pronoun, possessive, 3rd person plural	eiddynt
Rhacys1u	Rhagenw, cysylltiol person 1af unigol	Pronoun, conjunctive, 1st person singular	minnau
Rhacys2u	Rhagenw, cysylltiol, 2il person unigol	Pronoun, conjunctive, 2nd person singular	tithau
Rhacys3gu	Rhagenw, cysylltiol, 3ydd person gwrywaidd unigol	Pronoun, conjunctive, 3rd person masculine singular	yntau
Rhacys3bu	Rhagenw, cysylltiol, 3ydd person benywaidd unigol	Pronoun, conjunctive, 3rd person feminine singular	hithau
Rhacys1ll	Rhagenw, cysylltiol, person 1af lluosog	Pronoun, conjunctive, 1st person plural	ninnau
Rhacys2ll	Rhagenw, cysylltiol, 2il person lluosog	Pronoun, conjunctive, 2nd person plural	chithau

Tag	**Disgrifiad**	**Description**	**Enghraifft/example**
Rhacys3ll	Rhagenw, cysylltiol, 3ydd person lluosog	Pronoun, conjunctive, 3rd person plural	hwythau
Rhagof	Rhagenw, gofynnol	Pronoun, interrogative	pa, pwy, ble, beth
Rhadangg	Rhagenw, dangosol, gwrywaidd	Pronoun, demonstrative, masculine	hwn
Rhadangb	Rhagenw, dangosol, benywaidd	Pronoun, demonstrative, feminine	hon
Rhadangd	Rhagenw, dangosol, diryw	Pronoun, demonstrative, neutral	hyn
Rhaperth	Rhagenw, perthynol	Pronoun, relative	a, y
Rhaatb	Rhagenw, atblygol	Pronoun, reflexive	hun, hunain
Rhacil	Rhagenw, cilyddol	Pronoun, reciprocal	gilydd
Uneg	Unigryw, geirynnau negyddol	Unique, negative particles	dim, ni
Ucad	Unigryw, geirynnau cadarnhaol	Unique, affirmative particles	mi, fe
Ugof	Unigryw, geirynnau gofynnol	Unique, interrogative particles	a
Utra	Unigryw, geiryn traethiadol	Unique, predicative particle	yn
Ebych	Ebychiad	Interjection	oo, aa, hei
Gwest	Gweddilliol, estron	Other, foreign word	anyway
Gwfform	Gweddilliol, fformiwla	Other, formula	9 + x = y, E = mc2
Gwsym	Gweddilliol, symbol	Other, symbol	€
Gwacr	Gweddilliol, acronym	Other, acronym	GIG
Gwtalf	Gweddilliol, talfyriad	Other, abbreviation	Cyf.
Gwdig	Gweddilliol, digid	Other, digit	1, 99, 2000
Gwllyth	Gweddilliol, llythyren	Other, letter	A, B, Ch, a, b, ch
Gwann	Gweddilliol, annosbarthedig	Other, unclassified	
Atdt	Marc atalnodi, terfynol	Punctuation mark, final	. ? !
Atdcan	Marc atalnodi, canolig	Punctuation mark, medial	, ; :, –
Atdchw	Marc atalnodi, chwith	Punctuation mark, left	([
Atdde	Marc atalnodi, de	Punctuation mark, right	)]
Atdcys	Marc atalnodi, cysylltnod	Punctuation mark, hyphen	-
Atddyf	Marc atalnodi, dyfynnod	Punctuation mark, quotation	“” ‘ ’