LaTeX for Linguists

A short introduction to Linguistics packages

St John's Linguistics Society & CamTeX Society

Download here: https://github.com/nuria-bosch/LaTeX-for-Linguists

2 February 2022

Contents

Acknowledgements

Part of this workshop was significantly inspired by a previous one given by Brandon Papineau and Nina Markl at ULAB's 2021 Annual Conference, to whom acknowledgement is due. Their documentation goes into greater depth on Linguistics packages than the pages that follow, so we encourage you to check it out here:
`https://github.com/BranPap/LaTeX_for_Linguists`.

We have also made use of some information in this LaTeX4Ling webpage:
`https://www1.essex.ac.uk/linguistics/external/clmt/latex4ling/`

Any remaining issues or unclear presentation, our own.

1 Introduction

LaTeX is a type-setting language based on the programming language TeX. It is a software for document preparation. Unlike Microsoft Word, LaTeX isn't a "What You See Is What You Get" word processor in that it uses *plain* text as its source text. Formatting of the text itself is done via markup conventions (e.g., commands) to define the general structure and style of your document. You provide the content, the formatting indications and LaTeX converts it into a PDF.

LaTeX is widely used in academia and for the publication of scientific documents in many fields, such as mathematics, computer science and linguistics.

There are a number of LaTeX packages available for writing linguistics papers. Many have been created to generate numbered examples, syntactic trees, Optimality Theoretic tableaux, matrices, IPA symbols, among others. Certain packages are used standardly throughout the field (e.g., `tipa`), while others will vary depending on personal preferences.

Some packages:

- Glosses: `gb4e`
- IPA symbols: `tipa`
- Phonological rules: `phonrule`
- Syntax trees: `qtree`
  - `tikz-qtree` has the same syntax as qtree, but includes PGF/TikZ, which allows you to draw arrows (for example, for movement), among other things.
- Optimality Theory Tableaux: `OTtablx`

For simplicity, we will cover `qtree, tipa, phonrule, gb4e` here. You can browse additional packages here: `https://ctan.org/topic/linguistic`.

2 Phonetics and phonology

2.1 tipa

With the `tipa` package, inserting IPA (International Phonetic Alphabet) symbols into your documents is very easy.

In the preamble add:

```
\usepackage{tipa}
```

You can now input phonetic characters via their tipa names, as commands. For example, for a glottal stop /ʔ/, type `\textglotstop`. Knowing which commands correspond to which IPA symbols is not always predictable and is, for the most part, a matter of consulting documentation or memorisation (e.g., the lateral fricative /ɬ/ is coded as `\textbeltl`). You can view the full list of phonetic symbols (including diacritics) in the tipa documentation: `https://www.tug.org/tugboat/tb17-2/tb51rei.pdf`.

The above works fine if you only need to type one or two phonetic symbols. However, if you want to type something more complicated such as [ɛksplə'neɪʃən], it is much more straightforward to use "shortcut" characters. You could achieve the same result with individual commands, but the process would be messy. There are various ways of using shortcuts.:

textipa and tipaencoding

With the command `\textipa{}`, you can make the most of shortcuts. For example, `\textipa{[Ekspl@"neIS@n]}` produces [ɛksplə'neɪʃən]. Common shortcuts, such as the ones used in this example, are shown below. Again, these shortcuts are not fully predictable, so check the full documentation of tipa if the symbols you are looking for are not listed here:

ASCII	:	;	"							
TIPA	ː	ˑ	ˈ							
ASCII	0	1	2	3	4	5	6	7	8	9
TIPA	ʉ	ɨ	ʌ	ɜ	ɥ	ɐ	ɒ	ɤ	ɵ	ɘ
ASCII	@	A	B	C	D	E	F	G	H	I
TIPA	ə	ɑ	β	ɕ	ð	ɛ	ɸ	ɣ	ɦ	ɪ
ASCII	J	K	L	M	N	O	P	Q	R	S
TIPA	ɟ	ʁ	ʎ	ɱ	ŋ	ɔ	ʔ	ʕ	ɾ	ʃ
ASCII	T	U	V	W	X	Y	Z	|		
TIPA	θ	ʊ	ʋ	ɯ	χ	ʏ	ʒ	|		

For a useful cheatsheet with all the shortcuts for `\textipa{}` see here: `https://jon.dehdari.org/tutorials/tipachart_mod.pdf`.

An equivalent command is `{\tipaencoding{}}`. The main difference with textipa is that the command itself needs to be embraced by curly brackets. So the command below

```
{\tipaencoding{[Ekspl@"neIS@n]}}
```

produces [ɛksplə'neɪʃən]

IPA environment

Finally, another way you can add shortchuts is by means of the IPA environment. The shortcuts work in the same manner as `tipaencoding` or `textipa`, but it is type-set with a `\begin{}` environment:

```
\begin{IPA}
[Ekspl@"neIS@n]
\end{IPA}
```

2.2 PhonRule

`PhonRule` is a package that allows the creation of phonological rules. To use this package in your document, load the package in your preamble with `\usepackage{phonrule}`.

A basic phonological rule can be written with the command `\phon{/x/}{[y]}`, where 'x' here is the phoneme undergoing phonological change and 'y' is the surface phonological form after the change has applied. In order to add IPA symbols, we will of course have to make use of what you have learned in the section about the `tipa` package. For example, let's try to draw the spirantisation rule in Spanish and Catalan, whereby voiced plosives undergo spirantisation into voiced approximants in intervocalic contexts. We start with the command we have just introduced:

`\phon{/b/}{[\textbeta]}`

/b/ → [β]

We still need to know how to incorporate a conditioning environment however: underlying /b/ does not become [β] in all contexts in neither Catalan nor Spanish. We will do this by adding another set of curly brackets after the command `\phonc`, which will denote the environment: `\phonc{/x/}{[y]}{environment}`. If you need to add _ signs for the context, you should use the commands `\phonl`, `\phonr` and `\phonb` to add a _ sign on the right, on the left or on both sides, respectively (note that this is a bit counterintuitive: `\phonl` does not insert an underscore to the *left* of the phoneme, but rather to its right; and vice versa for `\phonr`.). For the Catalan/Spanish spirantisation rule, since the rule applies between two vowels, we will need _ on both sides. Therefore, we will input `\phonb{/b/}{[\textbeta]}{V}{V}`:

/b/ → [β] / V_V

Phonologically speaking, this may still be slightly unsatisfactory though: the rule applies to *all* voiced plosives, so /g/ and /d/ are being left out here. We could either add them inside the rule itself with commas

`\phonb{/b, d, g/}{[\textbeta, \textipa{D}, \textgamma]}{V}{V}`:

/b, d, g/ → [β, ð, ɣ] / V_V

but another option would be to incorporate phonological features into our LaTeX document, instead of enumerating phonemes individually. You can either add them manually with brackets inside the rule above (e.g., adding `[+cons, -son, +voice]`), which would generate:

`\phonb{[+cons, -son, +voice]}{[+cont]}{V}{V}`

[+cons, -son, +voice] → [+cont] / V_V

There is also a more elegant presentation style to incorporate phonological features, however. For this, you can use `\phonfeat{}` and add your features inside the command:

```
\phonb{\phonfeat{+cons \\ -son \\ +voice}}
{\phonfeat{+cons \\ -son \\ +voice \\ +cont}}{V}{V}
```

This code generates the following formatting for our spirantisation rule:

$$\begin{bmatrix} \text{+cons} \\ \text{-son} \\ \text{+voice} \end{bmatrix} \rightarrow \begin{bmatrix} \text{+cons} \\ \text{-son} \\ \text{+voice} \\ \text{+cont} \end{bmatrix} \text{ / V\_V}$$

3 Syntax

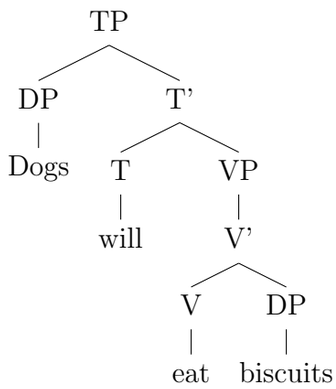
3.1 qtree

A simple and intuitive package for drawing syntax trees in LaTeX is `qtree`. This allows you to build syntactic trees in LaTeX. Equally, it can also be used to draw other kinds of trees (e.g., decision trees or other types of hierarchically-organised information). In order to implement it in your document declare the following command at the start of your document, i.e., in your preamble:

```
\usepackage{qtree}
```

To draw syntactic trees, start with the command `\Tree` and then continue by annotating your tree with bracketing notation, as shown below. Remember to add a period before each node label. Also remember to leave a space after the last word within a given group and before the actual bracket, as below:

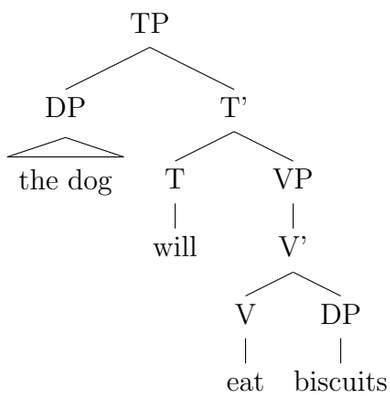
```
\Tree [.TP [.DP Dogs ] [.T' [.T will ] [.VP [.V' [.V eat ]
[.DP biscuits ] ] ] ] ]
```



If, however, you want to embed multiple words under the same node (e.g., [DP the dog]), you can make use of the `\qroof{}` command. Note, though, that in this scenario you will not to specify a bracketed group for the phrase and the label of the phrase will be specified at the very end of the command. For a DP containing *the dog*, the command would be as follows: `\qroof{the dog}.DP`

A fully fleshed sentence with the `\qroof{}` command would look like this:

```
\Tree [.TP \qroof{the dog}.DP [.T' [.T will ] [.VP [.V' [.V eat ]
[.DP biscuits ] ] ] ] ]
```



This short introduction will not cover movement, but you can use other commands that will allow you to strike-trough elements. Alternatively you can also add a trace.

To strike-trough a particular word or set of words, load the package `\usepackage{ulem}` and use the command `\sout{}`. For example, if you want to strike-through a wh-word such as *what*, follow this step after adding `\usepackage{ulem}` to your preamble:

```
\sout{what}
```

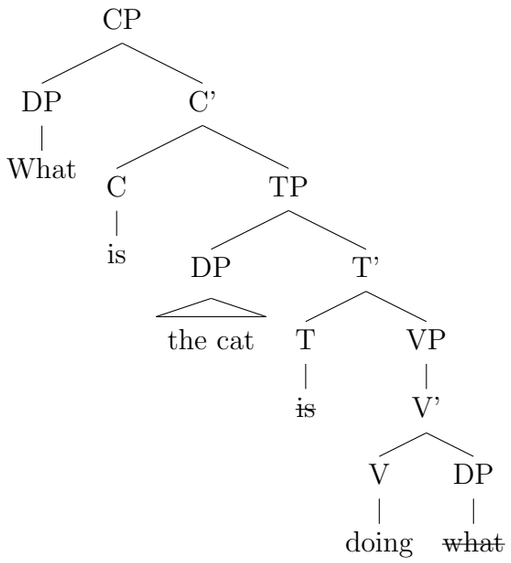
This will produce:

~~what~~

Traces can be added by adding a *t* with a subscripted index t_i, with `t$_i$` (the dollar sign in LaTeX is a way of entering math mode - specifically, inline math mode. What this means is that in order to enter any inline maths text, such as a subscript, you will have to embed the mathematical notation within two dollar signs).

Before finishing this section, let's apply this simplified way of drawing movement to an example:

```
\Tree [.CP [.DP What ] [.C' [.C is ] [.TP \qroof{the cat}.DP [.T'
[.T \sout{is} ] [.VP [.V' [.V doing ] [.DP \sout{what} ] ] ] ] ] ] ]
```



There are also related packages that function similarly, notably `tikz` and `tikz-qtree`. Since for the purposes of this short introduction they work comparably, we won't review them here. Have a look at `https://mirror.ox.ac.uk/sites/ctan.org/graphics/pgf/contrib/tikz-qtree/tikz-qtree-manual.pdf` for more information.

4 Morphology

4.1 gb4e

`gb4e` is a very handy Linguistics package for numbered examples and morphological glosses. As should be familiar by now, add `\usepackage{gb4e}` in your preamble. **Careful**, though: to be on the safe side, this should be the *last* package you will add to your document, otherwise it can seriously affect other packages.

Here is an example of what you can do:

(1) This package can create examples.

(2) This is the second example.
   a. You can also add subexamples.
   b. It keeps track of each numbered example, so it will update numberings if you delete examples or add new ones.

(3) * This example are ungrammatical.

The code for this particular snippet is the following one:

```
\begin{exe}
    \ex This package can create examples.
    \ex This is the second example.
    \begin{xlist}
        \ex You can also add subexamples.
        \ex It keeps track of each numbered example, so it will update numberings
        if you delete examples or add new ones.
    \end{xlist}
    \ex[*]{This example are ungrammatical.}
\end{exe}
```

Examples, therefore, are introduced by `\begin{exe}`. Remember, this has to be closed after all the examples with `\end{exe}`. Add each item with `\ex` and add subexamples by starting a `\begin{xlist}` and repeating the same procedure (remember to end this one too!).

If you would like to gloss examples such as a Russian sentence (say, *My s Marko poexali avtobusom v Peredelkino*), you'll divide your code into three main steps: (1) provide the data to be annotated, (2) the morphological gloss and (3) the translation (remember to add a line break after the first two steps with `\\`).

```
\begin{exe}
  \ex
  \gll  My s Marko poexa-l-i avtobus-om v Peredelkino \\
    1.PL COM Marko go-PST-PL bus-INS ALL Peredelkino \\
  \trans  'Marko and I went to Perdelkino by bus.'
\end{exe}
```

The code above generates the following:

(4) My s Marko poexa-l-i avtobus-om v Peredelkino
1.PL COM Marko go-PST-PL bus-INS ALL Peredelkino
'Marko and I went to Perdelkino by bus.'

4.1.1 Another possibility: `ExPex`

There are other packages that also type-set numbered examples and glosses. In the end, this is a matter of personal preference or for what purposes you are using the specific package, but some academic journals, for example, may prefer LaTeX submissions with other packages. One particular package you will likely also come across is `ExPex`, so we will briefly introduce an example here. Very quickly, let's "rephrase" the example provided in (3) with `ExPex` (of course, remember to add `\usepackage{expex}` in your preamble. This should be coded as follows.

```
\pex
  \begingl
  \gla My s Marko poexa-l-i avtobus-om v Peredelkino //
  \glb 1.PL COM Marko go-PST-PL bus-INS ALL Peredelkino //
  \glft 'Marko and I went to Perdelkino by bus.' //
  \endgl
\xe
```

This generates (as you will be able to tell, the resulting formatting and style of the gloss is slightly different to `gb4e`):

(1) *My s Marko poexa-l-i avtobus-om v Peredelkino*
1.PL COM Marko go-PST-PL bus-INS ALL Peredelkino

'Marko and I went to Perdelkino by bus.'

The main differences we can note here with the previous package is that we no longer use a `\begin{...}` command. Instead we start the numbered example with `\pex` and end it with `\xe`. The gloss is structured in the following way:

- `\begingl` informs the package to start a gloss.
- `\gla` is used for the sentence you want to gloss.
- `\glb` is used for the morphological gloss you provide.
- `\glft` is used for translation of the original sentence
- `\endgl` is used to end the gloss section.

Do **not** forget however to close `\gla`, `\glb` and `\glft` with two forward slashes `//`. Otherwise, the code will output an error and won't render properly.

5 Additional resources

- https://en.wikibooks.org/wiki/LaTeX/Linguistics
- https://psumikeputnam.weebly.com/latex-for-linguists.html
- https://adamliter.org/content/LaTeX/latex-workshop-for-linguists.pdf
- https://ctan.org/topic/linguistic?lang=en
- https://www1.essex.ac.uk/linguistics/external/clmt/latex4ling/