Example document for `org-linguistics.el`

Göktuğ Kayaalp

January 16, 2021

Abstract

`org-linguistics.el` is a simple Emacs Lisp package that defines some special blocks that help typeset syntax trees, glosses, and enumerated sentences. It's basically a dirty hack on top of Org mode's LaTeX export mechanism and the `org-element` API. This is also fresh-outta-oven very-WIP stuff, so use with care.

Trees

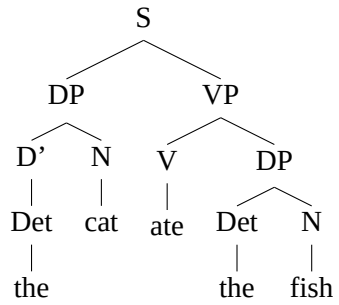
We use the `qtree` LaTeX package to generate syntactic trees from unordered lists and description lists. As seen in **??**, we write the tree analysis in a list form, and each level of the list corresponds to a level in the tree. If the node label has an apostrophe or a right single quotation mark at the end, it's parsed as a bar level. By default, it's exported as an apostrophe, but if you want the macron (overline) syntax, you can set `org-linguistics-barlevel-template` to `\=%s`.

```
#+name: testTree1
#+begin_tree
- S
  - DP
    - D’
      - Det
        - the
    - N
      - cat
  - VP
    - V
      - ate
    - DP
```

```
      - Det
        - the
      - N
        - fish
#+end_tree
```

Evaluating the above source code, we get a result like in (1):

(1)

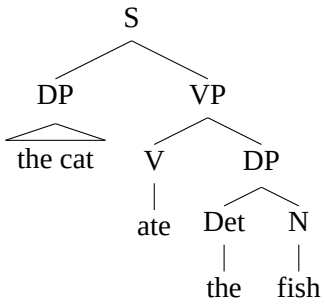


Another little shorthand that is available is to use description list items to generate “roofs” over phrases. Observe the line `DP :: the cat` in **??**.

```
#+begin_tree
- S
  - DP :: the cat
  - VP
    - V
      - ate
    - DP
      - Det
        - the
      - N
        - fish
#+end_tree
```

Observe then the first DP in (2):

(2)



This should cover the most usual and rudimentary syntactic trees. As it is now, for more advanced applications, directly using `qtree` or `tikz-qtree` packages would be more opportune.

Glosses

We use the `enumsentence` package to create glosses. An improvement `org-linguistics.el` provides over it is a naïve automatic line wrapping feature (viz. `org-linguistics-gloss-fill-column`

Glosses too are based on lists. In a `gloss` block we provide up to three lists: an `example` list that lists the constituents of the example sentence, a `gloss` list that lists a matching gloss, and finally a singleton `translation` list that lists a textual translation line.

The `example` list is mandatory. You can then either provide only one of `gloss` or `translation`, or provide both of them. The translation is not wrapped.

Any list item type should work, but it's advisable to use ordered lists as they help balance the glosses.

The examples below should make it clearer:

```
#+name: glossEx
#+begin_gloss
- example
  1) Ali
  2) topu
  3) at.
  4) Veli
  5) topu
  6) tut.
  7) Ayşe
  8) ip
  9) atla.
```

```
- gloss
  1) Ali-NOM
  2) ball-ACC
  3) throw-IMP
  4) Veli-NOM
  5) ball-ACC
  6) catch-IMP
  7) Ayşe-NOM
  8) rope-NOM
  9) jump-imp

- translation
  - Throw the ball, Ali. Catch the ball, Veli. Ayşe, jump rope.
#+end_gloss
```

The generated gloss figure:

(3) **Ali topu at. Veli topu tut.**
Ali-NOM ball-ACC throw-IMP Veli-NOM ball-ACC catch-IMP
'Throw the ball, Ali. Catch the ball, Veli. Ayşe, jump rope.'
Ayşe ip atla.
Ayşe-NOM rope-NOM jump-imp

Enumerated sentences

`enumsentence` can also be used to generate non-gloss enumerated sentences and sentence lists. In order to make use of this feature we use the `enum` blocks. If the block contains a single paragraph, it's exported as a single enumerated sentence:

```
#+name: enum1
#+begin_enum
The quick brown fox jumps over the lazy dog.
#+end_enum
```

Result:

(4) The quick brown fox jumps over the lazy dog.

We can obviously use the `enum1` label to refer to the sentence.

But if we include a list of sentences inside the `enum` block, then a sentence listing is generated, and each sentence has it's own label generated as a combination of the `enum` block name, a dash, and the zero-padded one-based two-digit index of the sentence, e.g. `enum2-01` (= 5a) and `enum2-02` (= 5b) for (**??**) as seen below:

```
#+name: enum2
#+begin_enum
- The quick brown fox jumps
- over the lazy dog.
#+end_enum
```

(5) a. The quick brown fox jumps

b. over the lazy dog.

Development and Contributions

`org-linguistics.el` was inspired by a post on r/emacs rather recently as of writing these lines (February 16th, 2021), so it is still a small, work-in-progress, malleable package. As it is now, captions and labels need better integrating as they may cause some breakage and don't work with internal links of Org mode.

Ideas for new blocks are most welcome, especially when they come with code (I'm only an MA student so not exactly an expert in linguistics notation yet at this point). Ditto for bug reports. You can submit pull requests and issues on the GitHub repository. Please read the relevant section on project README in order to learn the couple little requirement regarding commit messages.