ELAN-to-LaTeX 1.0 documentation

Eetu Sjöblom
firstname.lastname@helsinki.fi

1 Overview

ELAN-to-LaTeX (EtL) is a script for extracting interlinear glosses from ELAN annotations and using them in LaTeX documents. Given a ELAN file with the appropriate annotations, the script produces linguistic examples such as follows.

(1) *juoksen sinun taloosi*

juokse-n sinu-n talo-o-si
run-1SG you-GEN house-ILL-2SG.POSS

'I run into your house' [AB 10]

EtL has been developed to facilitate the handling of large amounts of examples by reducing the manual work involved in writing the necessary latex code. Each example receives its own LaTeX command which is constructed from the segment labels used in the ELAN annotation. The user only needs to know the label of the segment she wants to use as an example. If the user makes changes to the annotations, for example in the case of correcting typos, simply running the script again will update the LaTeX files as well.

EtL currently (version 1.0) only handles one specific ELAN template[1], supplied in the package with the script and this documentation. Future versions will handle arbitrary ELAN templates.

The script is written in Python (3.4+)[2] and uses the `gb4e`[3] package for the glosses. There are no additional dependencies; the script only uses standard Python libraries.

[1] Template courtesy of Nailya Philippova
[2] See `www.python.org` for more information and installation instructions
[3] `https://www.ctan.org/pkg/gb4e?lang=en`

2 Configuration

The script uses a configuration file `etl_conf`, where the user specifies the necessary filepaths and formatting options. At the moment the configuration file consists of three sections: *input files*, which are the ELAN files containing the annotations; *output file*, which contains the produced commands and functions as a LaTeX package; *small caps formatting*, where the user may specify which gloss elements she wants to print in SMALL CAPS (in the above example, 1SG, GEN, ILL, 2SG.POSS).

Filling in the configuration file is straight-forward and some instructions are found in the file itself. The file can be opened in any plain text editor.

3 Code

EtL has been written and tested on Python 3.4 and proper functioning on earlier versions is not guaranteed. The code is thoroughly commented and the potential developer should obviously refer to it for details on how the script works. Some general information is sufficient here.

The ELAN annotations are saved in a XML file, and the script extracts the information from the file using the standard Python `xml` package. More specifically, the `xml.etree.ElementTree` module[4], a simple and easy-to-use XML API, is used. The `ElementTree` class is a hierarchical data structure for representing the whole document. This suits well to the hierarchical structure of the ELAN tiers.

Potential developers find the code also in GitHub under the project name **elan-to-latex**.[5]

4 Using with LaTeX

Running the script results in a `.sty` file, which contains the LaTeX commands and functions as a package. It is used as any other package, with the `\usepackage` command. The file can be included in the directory with the LaTeX files, or in the appropriate package directory of your LaTeX installation. Also, as the `gb4e` package is used, it is necessary to include this too: `\usepackage{gb4e}`

The package consists of command definitions, each containing one liinguistic ex-

[4] See the Python documentation in `https://docs.python.org/` for more information
[5] `https://github.com/Eetusjo/elan-to-latex`

ample of the form mentioned above. The command names are derived from the ELAN segment names. For example, a segment called `AA_BB_15` will receive the command `\AABBOneFive`. Unfortunately the LaTeX commands can not contain digits or special characters. Consequently, the digits are represented as words (Zero, One, ..., Nine) and other non-alphabetical characters are simply omitted. This naming scheme is hoped to be the most intuitive possible and eliminate the need to ever refer to the package contents to find the needed command. The user only needs to know the correct segment she wants to use as an example.

The examples are updated by running the script again on the improved ELAN file. As long as the segment names are the same, the command names also stay unchanged.