ConlluEditor: a fully graphical editor for Universal dependencies treebank files

Johannes Heinecke
Orange Labs
2 rue Pierre Marzin
F - 22307 Lannion cedex
`johannes.heinecke@orange.com`

Abstract

In this paper we present the ConlluEditor annotation tool for manual annotation of files in CoNLL-U format, such as Universal Dependencies treebanks. Apart from providing a graphical editor for basic and enhanced dependencies, multi-token words, it also runs validation scripts to find potential errors. ConlluEditor uses a client-server architecture. It is freely-available under the 3-Clause BSD License.

1 Introduction

The Universal dependencies (UD) project (Nivre et al., 2016) currently contains more than 140 treebanks in nearly 80 languages, all annotated following the same guidelines for POS tags (UPOS) and dependency relations and stored in a standard format, CoNLL-U. Some treebanks have been converted from an original annotation scheme, whereas others have been annotated from scratch using the UD annotation guidelines. Annotating new sentences for a treebank with a text editor is diffcult, that is why many editing tools exist to help annotators.

2 Related Work

The UD project lists some of the most prominent editors for UD data[1]. We describe a subset of these editors, which we used for various annotation tasks, in order to explain the motivation for this work. We do not go into detail for annotation tools, which do not handle (import/export) CoNLL-U format.

BRAT (Stenetorp et al., 2012) is a powerful annotation tool, which can be configured in detail for manu different annotation tasks, like named entities, coreferences, POS tagging, and dependency relations. BRAT uses an internal character-based format, which can be transformed into CoNLL-U. Since the annotation is character-based, it is the annotator who can decide the start and the end of the tokens. BRAT uses a server and a frontend within a web browser. The display of the annotations, however, becomes confusing if the linked tokens are too wide apart to be shown on a single line.

WebAnno (Eckart de Castilho et al., 2016) is like BRAT a general annotation tool, as powerful as the former and highly configurable through a web interface. As BRAT WebAnno is multi-user and annotations can be made in parallel. apart from UD's CoNLL-U format, WebAnno reads and writes many other standard annotation formats, depending on the annotation tasks. As BRAT, WebAnno displays dependency trees in a flat graph mode (cf. figure 3). Long sentences are wrapped into multiple line, which makes it sometimes difficult to grasp complex dependencies.

UD Annotatrix (Tyers et al., 2018) is a lightweight browser based annotation tool for UD treebanks. Dependency trees can either be edited in a graphical (flat graph) mode or by directly modifying the underlying CoNLL-U data. Annotatrix provides tools to modify the tokenisation (splitting or joining tokens). Apart from CoNLL-U other formats can be used, including plain text. Dependency graphs can be exported as image or LaTeX-code.

Arborator (Gerdes, 2013) permits, like WebAnno, a collaborative annotation of dependency corpora. As Annotatrix, the dependency graphs can be edited in a graphical mode or by modifying the underlying

[1] https://universaldependencies.org/tools.html

CoNLL-U code. Like WebAnno, the corpus being annotated is curated by an administrator. Several annotations of a single sentence can be compared to find the best annotation. Arborator provides a complex search language which enables the annotator to find examples of existing annotations. The dependency graphs can be exported in several image formats. Both Arborator and Annotatrix display sentences as graphics, without wrapping into multiple lines.

3 Motivation

Why yet another tool? Every tool has advantages and inconveniences, either technical issues (online/offline, needs a server or not, provides a functionality like searching or validating or not) or ergonomic ones (size of graphs/trees displayed, edition mode).

When we started working with UD treebanks, the then existing tools to graphically display dependency trees did not provide all display and search functions we needed. Further, for a demo we needed an offline solution. Finally, from a personal point of view, flat dependency graphs are more difficult to understand than dependency trees, which show horizontally the dependents. Notably in long sentence, one can see quickly the clausal structure. What started as a quick javascript hack to display dependency trees graphically, grew finally into a fully fledged editor which we tested in correcting existing CoNLL-U files and annotations of new sentences from scratch. Annotating dependency relations and correcting token information like form, lemma UPOS, XPOS and features proved to be really fast with this tool.

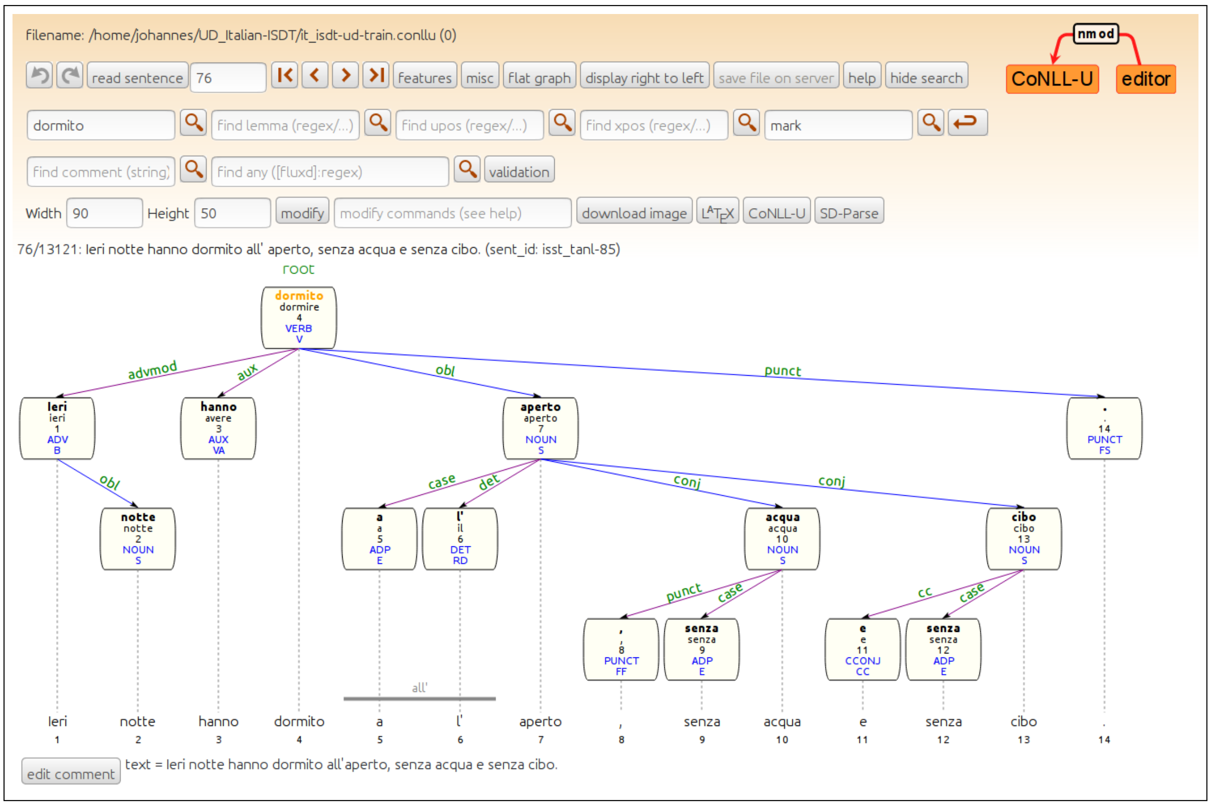
ConlluEditor provides the following features, which will be explained in the remaining sections:

- full graphical editor for basic and enhanced dependency relations
- word edit (form, lemma, UPOS, XPOS, features, misc-column)
- autocompletion (UPOS, XPOS, deprels; lists of valid labels must be provided)
- editing multitoken words (`[1-2] ...`)
- support for empty nodes (`[5.1] ...`)
- comment editing
- support for right-to-left scripts like Arabic or Hebrew
- split and join words (to correct bad tokenization)
- split and join sentences (to modify sentence segmentation)
- undo/redo
- regex search functions (including sequences of tokens, sub-graphs and comments)
- git support (add/commit)
- validation: indicates undefined UPOS, XPOS, dependency relations (based on lists given to the server)
- prohibition of invalid (cyclic) trees
- normalisation of token ids (first column, from 1 to n, taking into account multitoken words, empty words and heads)
- validation with external script (such like UD's `validate.py`[2]) on the current sentence
- export of dependency graphs as `.svg` image, LaTeX-code (for the tikz-dependenciy[3] package or the `deptree.sty`[4] LaTeX style, provided by ConlluEditor), sd-parse[5], CoNLL-U
- limited multi-user support: as long as two users do not edit the same sentence

ConlluEditor does not (yet) allow collaborative working like Arborator or WebAnno or other tools. So each annotator works on an individual copy which have to be merged in a second step.

The ConlluEditor is under the 3-Clause BSD License[6] and available at https://github.com/Orange-OpenSource/conllueditor.

[2]https://github.com/UniversalDependencies/tools.git
[3]https://ctan.org/pkg/tikz-dependency
[4]https://github.com/Orange-OpenSource/conllueditor/blob/master/doc/deptree-doc.pdf
[5]http://nlp.stanford.edu/software/stanford-dependencies.shtml
[6]https://opensource.org/licenses/BSD-3-Clause


Filename: /home/johannes/UD_Italian-ISDT/it_isdt-ud-train.conllu (0)

read sentence 76

dormito

mark

Width 90 Height 50

76/13121: Ieri notte hanno dormito all' aperto, senza acqua e senza cibo. (sent_id: isst_tanl-85)

text = Ieri notte hanno dormito all'aperto, senza acqua e senza cibo.

Figure 1: Main view in tree mode (note the multitoken word at nodes 9 and 10). The search buttons can be hidden to have place for deeper trees. Morpho-syntactic features can be displayed or hidden (to gain vertical space). Search results are highlighted (here *dormito*). The horizontal and vertical spacing can be modified to accomodate longer sentences. In order to change the font size, a `.css` style sheet can be modified. A help-Button provides detailed help on how to use the editor interface. (Example taken from the UD Italian-ISDT treebank).

4 Features

ConlluEditor is a server/client architecture with a server implemented in java which reads and manages the CoNLL-U file, writes modifications to disk and commits the changes to a git repository (if under git control). The front-end is implemented in javascript (using Bootstrap and jQuery), the main edit view is shown in figure 1. The data is passed between client and server using AJAX. The front-end has been tested with recent versions for Firefox, Chrome and Edge on Linux and Windows 10.

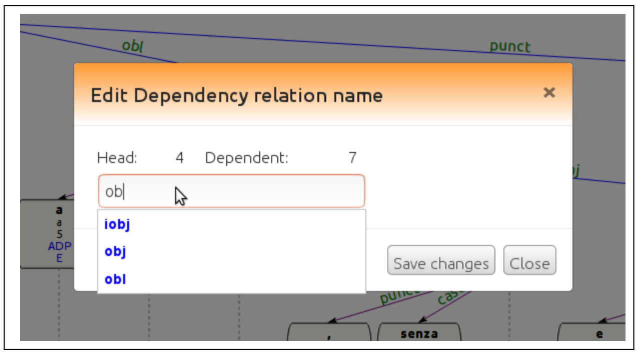
4.1 Graphical editing

Annotating dependency relations graphically speeds up the annotation work enormously. In ConlluEditor a simple click on a word (the future dependent) and a subsequent click on a second word (head) establishes a dependency relation, a dependency label edit window (fig. 2) opens to set the dependency relation label. Clicking twice on the same word makes this word root.

The graphical display of dependency trees is often a matter of personal taste. For space reasons, dependency graphs are often presented as flat graphs (e.g. generated with the tikz-dependencies package for LaTeX). In order to visually understand the syntax of a tree, a tree presentation (cf. figure 1) is often clearer. ConlluEditor proposes both views (check-button). Figure 3 shows the flat mode (here, the search functions are hidden).

In order to modify the word, a double-click or Control-click opens the word edit window (fig. 4). In order to avoid errors on UPOS, XPOS or dependency labels, ConlluEditor reads lists of valid labels and provides autocompletion. It also highlights invalid values in the main view.

Multitoken words can be edited (first and last word, multitoken wordform), by clicking on the multi-

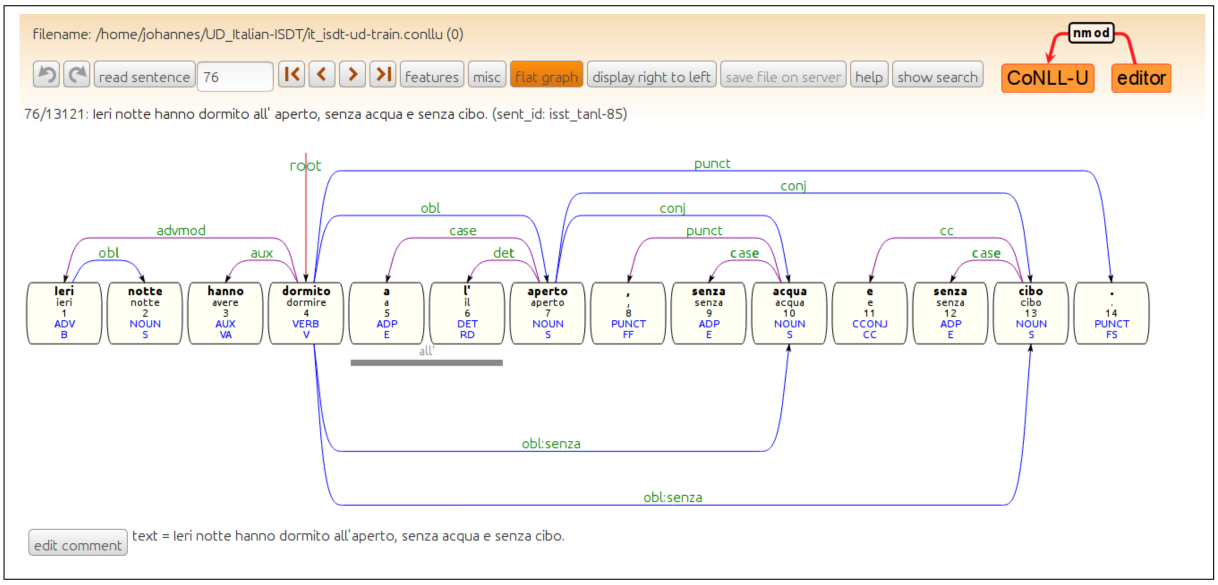

Edit Dependency relation name

Head: 4 Dependent: 7

obl

Figure 2: Dependency label edit window. All possible labels are proposed via autocompletion. Invalid labels, however, are accepted.

token indicator.


Filename: /home/johannes/UD_Italian-ISDT/it_isdt-ud-train.conllu (0)

76/13121: Ieri notte hanno dormito all' aperto, senza acqua e senza cibo. (sent_id: isst_tanl-85)

text = Ieri notte hanno dormito all'aperto, senza acqua e senza cibo.

Figure 3: Dependency tree in flat mode. Multitoken words are marked as in tree mode. In flat mode, enhanced dependencies are also displayed and can be edited.

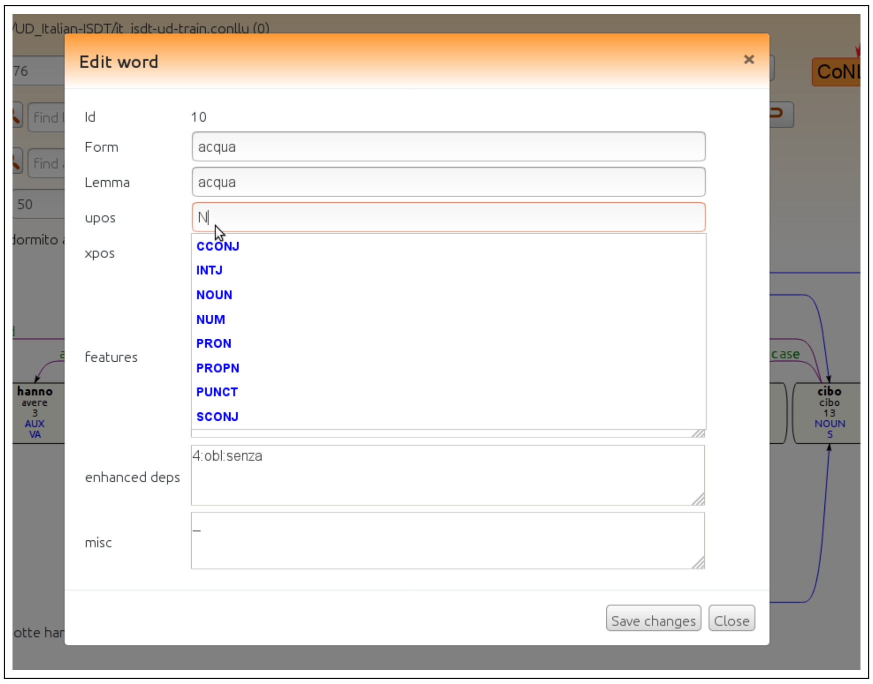
4.2 Searching

Once treebanks grow in size, search for similar phrases becomes extremely useful, not only to annotate similar structures in an similar way. ConlluEditor provides searching (with regular expressions) for forms, lemmas, UPOS, XPOS, dependency relations and comments. Searching sequences of forms, lemmas etc. is also possible, as well as a combination of these, like searching for all UPOS `PRON` which precede the lemma *fish*. It is, however, much less powerful than, for example, the Grew system (Bonfante et al., 2018)[7], which provides a complex query language to find sub-trees/sequences of forms, lemmas, UPOS, etc.

4.3 Data export and *git* support

Currently ConlluEditor reads and saves only files in the CoNLL-U standard[8]. However the interface permits to get the current sentence either as an `.svg` image or as code for LaTeX (tikz-dependencies), sd-parse or CoNLL-U. A simple click on the corresponding button opens a window to copy the generated code. The LaTeX-code includes enhanced dependencies.

[7]http://match.grew.fr/
[8]http://universaldependencies.org/format.html


Edit word

Id	10
Form	acqua
Lemma	acqua
upos	N
xpos	
features	
enhanced deps	4:obl:senza
misc	_

Figure 4: Word edit window (proposed a second method of editing enhanced dependencies). For UPOS, XPOS and dependency relation labels, the editor proposes autocompletion of valid values.

If the edited CoNLL-U file is under git version-control, ConlluEditor performs a *git add* and *git commit* on every edit. The commit message contains information about the document, and the modified word and sentence. In order to have less commits, a server option permits to have commits only after *n* edits.

4.4 Display modes

Apart from the two layout modes (tree and flat), ConlluEditor supports languages written in right-to-left mode like Arabic or Hebrew (figure 5).

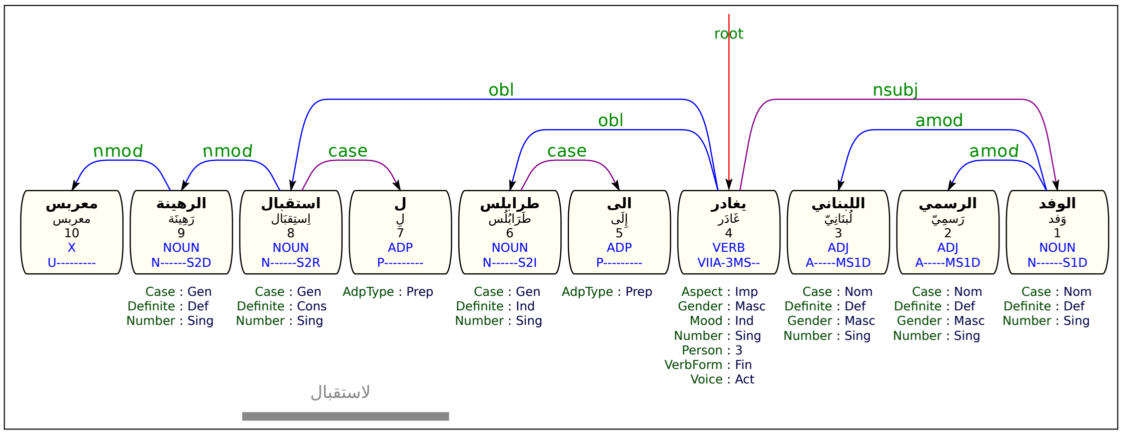


Figure 5: Support for languages which write right-to-left (example taken from the UD Arabic-PADT treebank). Here the display of morphological features is activated.

Whereas enhanced dependencies are currently only shown in flat mode, empty nodes (using the `[5.1]` format in the ID column) are shown in both modes (cf. fig. 6).

4.5 Validation of annotated sentence

ConlluEditor can run any validation programme on the current sentence. A validation script and its arguments must be defined in a configuration script which is given to the ConlluEditor server. Clicking

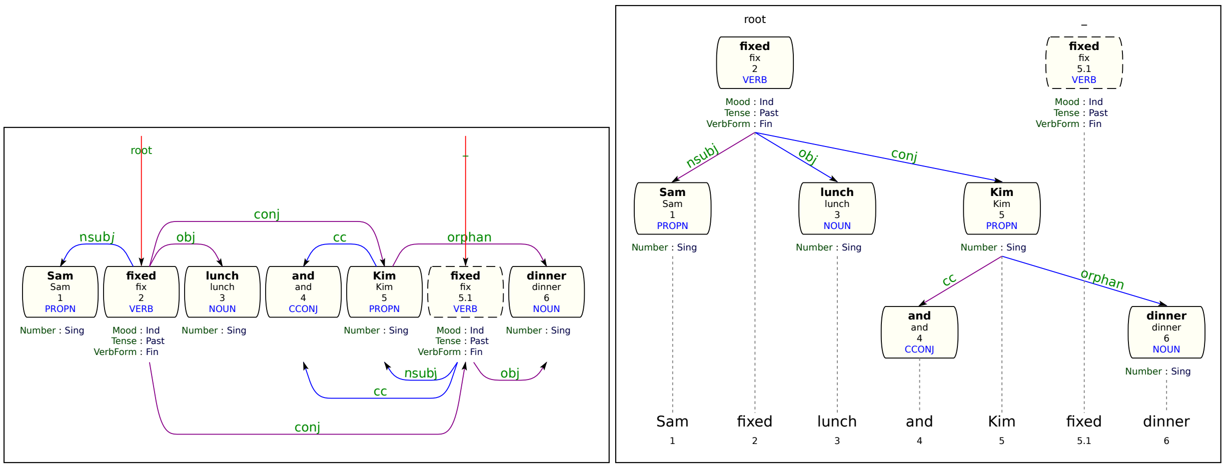
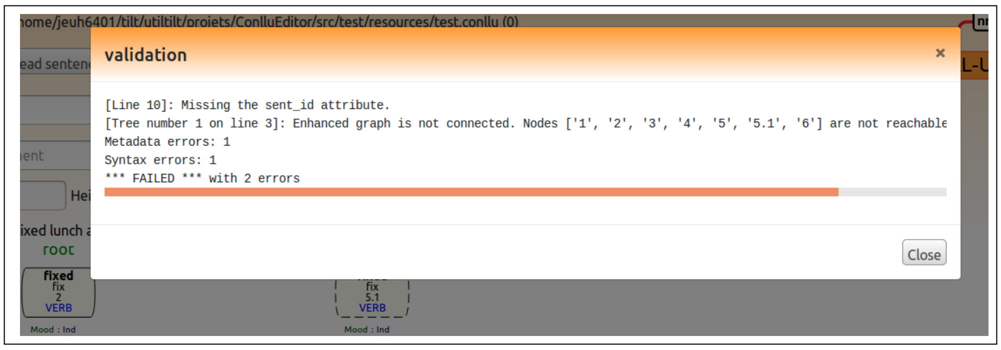


Figure 6: Display of empty nodes in both modes (also showing morphological features)

on the validation button runs the script on the current sentence and shows the entire output in a window (fig. 7).



```
[Line 10]: Missing the sent_id attribute.
[Tree number 1 on line 3]: Enhanced graph is not connected. Nodes ['1', '2', '3', '4', '5', '5.1', '6'] are not reachable
Metadata errors: 1
Syntax errors: 1
*** FAILED *** with 2 errors
```

Figure 7: Output of the validation programme

5 Future work

The main features we plan to implement is support for CoNLL-U Plus (`.conllp`) files[9] and finally add (language specific) validation rules, to find formal errors or inconsistencies, cf. also Marneffe et al. (2017). Even though CoNLL-U has become a standard for syntax annotations, other formats exists. ConlluEditor currently exports three formats (`conllu`, `sd-parse` and `latex`), others can be implemented rapidly if needed.

References

Guillaume Bonfante, Bruno Guillaume, and Guy Perrier. 2018. *Application of Graph Rewriting to Natural Language Processing*. Wiley-Iste.

Richard Eckart de Castilho, Éva Mújdricza-Maydt, Seid Muhie Yimam, Silvana Hartmann, Iryna Gurevych, Anette Frank, and Chris Biemann. 2016. A Web-based Tool for the Integrated Annotation of Semantic and Syntactic Structures. In *Proceedings of the International Conference on Computational Linguistics COLING*, pages 76–84.

Kim Gerdes. 2013. Collaborative Dependency Annotation. In *Proceedings of the Second International Conference on Dependency Linguistics (DepLing)*, pages 88–97, Prag.

[9]`http://universaldependencies.org/ext-format.html`

Marie-Catherine de Marneffe, Matias Grioni, Jenna Kanerva, and Filip Ginter. 2017. Assessing the Annotation Consistency of the Universal Dependencies Corpora. In *Proceedings of the Fourth International Conference on Dependency Linguistics*, pages 108–115, Pisa, Italy.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Yoav Goldberg, Jan Hajič, Manning Christopher D., Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *the tenth international conference on Language Resources and Evaluation*, pages 23–38, Portorož, Slovenia. European Language Resources Association.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. BRAT: a Web-based Tool for NLP-Assisted Text Annotation. In *13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107.

Francis M. Tyers Tyers, Mariya Sheyanova, and Jonathan North Washington. 2018. UD Annotatrix: An annotation tool for Universal Dependencies. In *ACL 2018*, pages 10–17, Melbourne.