Procesamiento del Lenguaje Natural, Revista nº 61, septiembre de 2018, pp. 185-188 recibido 26-03-2018 revisado 19-04-2018 aceptado 03-05-2018

TEITOK as a tool for Dependency Grammar

TEITOK y gramática de dependencia

Maarten Janssen
CELGA-ILTEC, University of Coimbra, Largo da Porta Férrea, Coimbra, Portugal
maartenjanssen@uc.pt

Abstract: TEITOK is an online platform for visualizing, searching, and editing TEI-based corpora. TEITOK has a modular set-up, and amongst other things provides an interface to easily add dependency relations to a tokenized TEI document, and provides the option to visualize the dependency trees, edit them, and search through the corpus using dependency relations.
Keywords: TEI/XML, dependency grammar, CWB

Resumen: TEITOK es una herramienta *online* para visualizar, buscar y editar corpora en el formato TEI/XML. TEITOK utiliza un sistema modular y, entre otras funcionalidades, contiene la opción de añadir dependencias a un documento tokenizado en TEI. También permite visualizar las dependencias de árbole, editar los errores y buscar el corpus resultante utilizando las dependencias.
Palabras clave: TEI/XML, gramática de dependencia, CWB

1 *Introduction*

TEITOK (Janssen, 2016) is an online platform for visualizing, searching, and editing TEI-based corpora. Other than text-based corpora, corpora encoded in the TEI/XML language (Consortium,) can keep all the typographic information that the original document contains, with a very rich set of annotations defined to account for the demands of very different types of corpora. This makes TEI the preferred representation format for corpora where detailed annotation is crucial, for instance for historical corpora (where detailed paleographic annotation is essential), and learner corpora (where annotation of corrections and errors by the student are of primary importance). But adding linguistic annotations to TEI documents is notoriously difficult. TEITOK attempts to remedy this by providing a user-friendly interface to manipulate TEI documents, tokenize them inline and run computational tools on the document behind the scenes.

TEITOK has a modular design that allows a different visualization of the documents in the corpus depending on their content. There are for instance specific visualization and editing modules for manuscript-based documents, time-aligned spoken documents, and interlinear glossed texts. All these modules work with the same base XML files, and can hence be combined in a single corpus, or even for a single file that contains, say, both manuscript data and a morphological analysis.

In this demo we will present the modules that TEITOK incorporates to work with dependency grammar: there is a script that allows you to add dependency relations to tokenized TEI/XML documents, and then visualize the resulting dependency trees as an SVG image. Incorrect trees can be edited directly in the interface. And you can search through a dependency-parsed corpus in a modified version of the CWB Corpus Query Language that allows you to access the head of a token directly. These different modules make TEITOK a powerful online platform for working with dependency grammars, and do so in an interface that can combine the use of dependency grammar with corpora containing rich typographic information, sound data or a morphological analysis.

1.1 File format

A corpus in TEITOK is a collection of files in a TEI/XML compliant format, with some minor adaptations to make it suitable for use in corpus linguistics. The documents can contain virtually any type of information TEI allows you to use, with few restrictions.

The way in which TEITOK adds linguistic annotations to existing documents in the TEI/XML format is by adding inline tokenization nodes, similar to the ⟨w⟩ nodes in standard TEI, and then add linguistic annotations as attributes over those nodes. Since TEITOK takes a linguistic perspective in which punctuation marks are considered tokens, token nodes are named ⟨tok⟩ instead of the standard ⟨w⟩ in TEI/XML, and the possible attributes of ⟨tok⟩ are much more liberal than what standard TEI/XML allows.

For dependency relations, TEITOK uses the columns and names of the CoNLL-U format, converted into attributes over the ⟨tok⟩ elements. An example of a simple two-word dependency-parsed sentence *Good day* in the TEITOK format is given in Figure 1.

Working with the TEITOK format is not that different from working with a line-based format like CoNLL, except that tokens are sequences of XML nodes rather than sequences of lines. However, a TEI-based corpus maintain much more of the information from the source document: in TEI/XML it is easy to keep inter-token spacing, which has to be done in a circumvent way in a line-based format. And it is possible to keep any typesetting information, such as paragraphs, bold and italic text, deleted and added text, color, footnotes, etc. interspersed among the ⟨tok⟩ elements.

It is easy to convert the output of a dependency parser to the TEITOK format, but that would not keep any typesetting data. Therefore, TEITOK assumes you create TEI/XML documents using any of the dedicated tools out there, including some provided by TEITOK itself, and then tokenizes the resulting document. For the tokenized document, it then provides the option to run a parser on the text automatically. This will export the list of tokens to a verticalized format, in which it keeps the ID of each word together with the word itself. It then runs the verticalized text through the web-service of the UDPIPE (Straka and Straková, 2017) multilingual dependency parser. And finally it imports the resulting parsed data back into the XML using the ID of each resulting token. Although this process by default uses UDPIPE, the process can be easily adapted to use any dependency parser using the CoNLL-U format.

2 *Visualization and Editing*

Once dependency relations have been added to the XML document, TEITOK can produce a dependency graph for each sentence in the document, either as a sentence with arches or as a dependency tree. The images are drawn as SVG images, which can be downloaded as SVG, or as rasterized PNG images for easy inclusion in publications.

Since a sentence in TEITOK can contain not only the dependency information but also a lot of typographic information, the sentence itself is visualized above the graph. In TEITOK, this visualization is done by simply reproducing the raw XML of the ⟨s⟩ node in the HTML document, and use CSS to display the TEI mark-up in a visually intuitive way. And since each token in TEITOK can contain much more information than just the dependency relations, all additional attributes on the ⟨tok⟩ (such as the POS tag and the lemma) are shown in a roll-over popup when the mouse is moved over a token in the sentence or in the tree. And to make it easy to see which word in the sentence corresponds to a given node in the tree, the word is highlighted when moving your mouse over a node in the tree. An example of the interface is shown in Figure 2, showing a sentence with the mouse hovering over the node *por* in the tree.

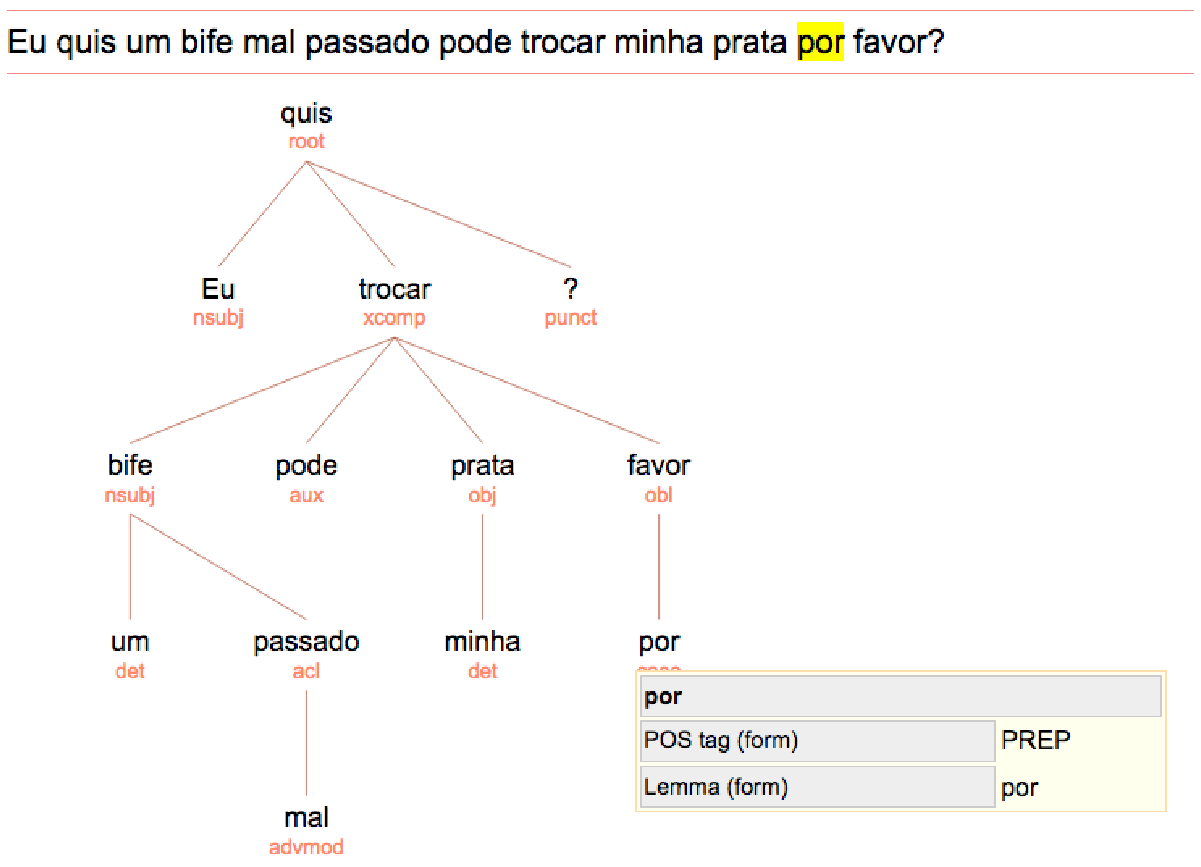
For corpus administrators, the same tree visualization can also be used to correct errors in the automatically produced dependency parses. This works in a very simple and intuitive way: while in edit mode, just click on the wrongly attached node, and then select where to reattach it. Doing so will redraw the graph and allow you to save the resulting tree back into the corpus. To correct an edge label, just click on the label and select the corrected label from a pop-up list, which will by default show the universal dependency labels, but can be customized to work with any other tagset as well. With these options, it is relatively quick to turn an automatically parsed corpus into a manually verified gold-standard corpus.

3 *Searching*

To make corpora searchable, TEITOK exports all tokens of all XML documents in the corpus to a CWB corpus (Evert and Hardie, 2011) using a custom encoding program called TT-CWB-ENCODE that di-

```
<TEI>
<teiHeader/>
<text id="test.txt">
<p id="p-1"><s id="s-1">
  <tok id="w-1" lemma="good" upos="ADJ" xpos="JJ"
    feats="Degree=Pos" head="w-2" deprel="amod">Good</tok>
  <tok id="w-2" lemma="day" upos="NOUN" xpos="NN"
    feats="Number=Sing" deprel="root">day</tok>
</s></p>
</text></TEI>
```

Figure 1: Example in TEITOK format



por	
POS tag (form)	PREP
Lemma (form)	por

Figure 2: A screenshot of the TEITOK dependency tree view

rectly reads XML files and writes CWB files. Since it is typically not desirable or possible to export everything in the XML file to the searchable corpus, the corpus administrator can define what from the XML files is exported to the CWB corpus, exporting XML regions as sattributes and token-attributes as pattributes.

TEITOK keeps a deep integration between the XML files and the CWB corpus: rather than showing the search results from the CWB query, TEITOK displays the corresponding XML fragments from the XML files used to build the CWB corpus. And the CQL query can also be used to modify selected tokens in the underlying XML files in an efficient manner.

Although CQL can be used to query a dependency-parsed corpus and allows you to search by word, lemma, POS, or edge label, it does not really allow you to search using the dependency relations themselves. TEITOK comes with a custom version of the query language called TT-CQP that does allow this.

```
(2) a:[word="prata" & deprel="obj"] :: head(a).upos="VERB";
    sort head(a).lemma; tabulate a[-1].word, head(a).lemma, head(a).upos
```

Figure 3: Search query in TT-CQP

TT-CQP uses the same files as CWB (and can hence be used with existing CWB corpora), and partially implements the CQL language, while adding a set of additional options to the language.

One added option is that in TT-CQP you can define restrictions on the head of a token, or access the head of a token directly in the output. An example of a query in TT-CQP doing this is given in Figure 3, which looks for all occurrences of *prata* (plate in Portuguese) in the corpus that have a verbal head. It then sorts the results on the lemma of the head, and outputs a list of the first word to the left of each occurrence, as well as the lemma and the POS of the head. For a corpus containing the sentence in Figure 2, that would hence include the result "minha trocar VERB".

Using a query very similar to that in Figure 3, TEITOK can produce an output close to a word sketch from the SketchEngine (Kilgarriff et al., 2014): the list of most typical heads and daughters of a word, ordered by their mutual information score. Such a list gives you the verbs that *prata* is most typically an argument of, or the adjectives that are most frequently used with *prata*.

4 Conclusion

TEITOK makes it easy to add dependency relations to existing corpora in the TEI/XML format, taking care of tokenization and parsing behind the scenes. It also makes it easy to correct trees in the interface, and does all this without requiring much understanding of the underlying computational processes from the corpus administrator. This should allow a lot more people to include dependency relations to their corpora, including corpora typically not treated this way such as historical corpora, spoken corpora, and learner corpora.

Once provided with dependency relations, TEITOK allows users to visualize and search the resulting dependencies parses in an intuitive way to make sure the resulting dependency parses are also usable for a wider audience. And it does all this in an interface that also provides a number of tools of a very different nature, such as tools specific to manuscript transcriptions, time-aligned spoken corpora, and morphologically parsed corpora. Hopefully this will lead to an increase in the use of dependency relations in a larger range of corpora.

References

Consortium, T. Guidelines for electronic text encoding and interchange. http://www.tei-c.org/P5/.

Evert, S. and A. Hardie. 2011. Twenty-first century corpus workbench: Updating a query architecture for the new millennium. In *Corpus Linguistics 2011.*

Janssen, M. 2016. TEITOK: Text-faithful annotated corpora. *Proceedings of the 10th International Conference on Language Resources and Evaluation, LREC 2016*, pages 4037–4043.

Kilgarriff, A., V. Baisa, J. Bušta, M. Jakubíček, V. Kovvář, J. Michelfeit, P. Rychlý, and V. Suchomel. 2014. The sketch engine: ten years on. *Lexicography*, pages 7–36.

Straka, M. and J. Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada, August. Association for Computational Linguistics.