UDWiki: guided creation and exploitation of UD treebanks

Maarten Janssen
Charles University
Faculty of Mathematics and Physics
Prague, Czechia
janssen@ufal.mff.cuni.cz

Abstract

UDWiki is an online environment designed to make creating new UD treebanks easier. It helps in setting up all the necessary data needed for a new treebank up in a GUI, where the interface takes care of guiding you through all the descriptive files needed, adding new texts to your corpus, and helping in annotating the texts. The system is built on top of the TEITOK corpus environment, using an XML based version of UD annotation, where dependencies can be combined with various other types of annotations. UDWiki can run all the necessary or helpful scripts (taggers, parsers, validators) via the interface. It also makes treebanks under development directly searchable, and can be used to maintain or search existing UD treebanks.

1 Introduction

There are many tools available to work with Universal Dependency (UD) - to annotate, visualize, edit, or processes treebanks. The majority of those tools are components only handling part of the construction of a treebank. This makes it necessary for people interested in creating a new treebank to read up on quite a few things about UD: on the principles behind the framework, on which documents are needed to submit a treebank, on which features would apply to the language in question, etc. Given the large amount of languages for which there is a UD treebank, all this is clearly not an insurmountable hurdle. But it likely makes that only people with a more computationally oriented profile will be attracted to create a treebank. And for less resourced languages (LRL), there might not always be computationally oriented linguists available.

This paper describes a more global environment for creating and maintaining UD treebanks, which aims to bring all the steps needed to create a new treebank into a single location, and which tries to make the process as easy as possible. This environment, called UDWiki, is a re-implementation of the by now defunct CorpusWiki (Janssen, 2016a), which was an attempt to provide a guided interface to allow people to build their own Part-of-Speech (POS) tagged corpus.

UDWIKI is built on top of the TEITOK (Janssen, 2016b) online corpus environment. TEITOK provides all the necessary tools to maintain (UD) treebanks, and UDWiki adds several specific tools to create treebanks for new languages from scratch, and make sure the treebank adheres to the UD standards. In its current state, UDWiki is fully operational, and has been tested on actual treebanks. But it is not yet a polished model - the idea behind the system is to fine-tine it based on the needs arising from the actual use of the system.

This paper describes the philosophy behind UDWiki and its implementation in TEITOK. It will describe UDWiki from the perspective it was initially conceived for - the creation of treebanks for new languages. And due to feedback from the reviewers it will also describe TEITOK as a generic tool for working with UD treebanks, and highlight the additional options the TEITOK environment brings over CoNLL-U based treebanks - mostly concerning mark-up that go beyond base UD annotation.

2 TEITOK and UD

TEITOK is an online environment for building, maintaining, publishing and searching linguistically annotated corpora. It was not initially designed for UD, and does not use the CoNLL-U format for its

internal storage. It rather stores corpus files in the TEI/XML format, where UD information can be embedded. TEITOK has incorporated several specific tool to work with UD treebanks (Janssen, 2018), and is used for the publication of a full version of the UD treebanks at LINDAT[1].

Dependency syntax in TEI is often represented in a manner quite unlike CoNLL-U, in which the morphology and lemma is kept on the word, while the dependency relations are kept as stand-off links elsewhere in the XML. An examples can be found in the SSJ500k corpus (Dobrovoljc et al., 2019). The approach in TEITOK is less strictly TEI, but much closer to CoNLL-U, and can be seen as a direct XML version of the CoNLL-U format: all tokens are modeled as ⟨tok⟩ nodes, the original text (form) is kept as the text content of that node, and all other columns in a CoNLL-U file are kept as attributes. So the difference in representation of tokens between TEITOK and CoNLL-U is small: instead of lines with columns there are nodes with attributes. An example of an annotated token in TEITOK is given in Figure 2, where the first 9 attributes correspond to the various columns in CoNLL-U, while the last two attributes (@id and @head) are specific to the TEITOK format. The two empty attributes (@deps and @misc) in this example are kept for clarity, but in TEITOK empty attributes would typically be suppressed.

```
<tok ord="3" lemma="word" upos="NOUN" xpos="NN" feats="Number=Sing"
  uhead="5" deprel="obl" deps="_" misc="_"
  id="w-5" head="w-3">Word</tok>
```

Figure 1: A Token in the TEITOK/XML representation

TEITOK is meant not only to publish, but also edit corpora. The editing of tokens is done in an intuitive way: from the running text, just click on a word. This will pop-up an HTML form as in figure 2, where all the attributes of the token can be directly corrected.

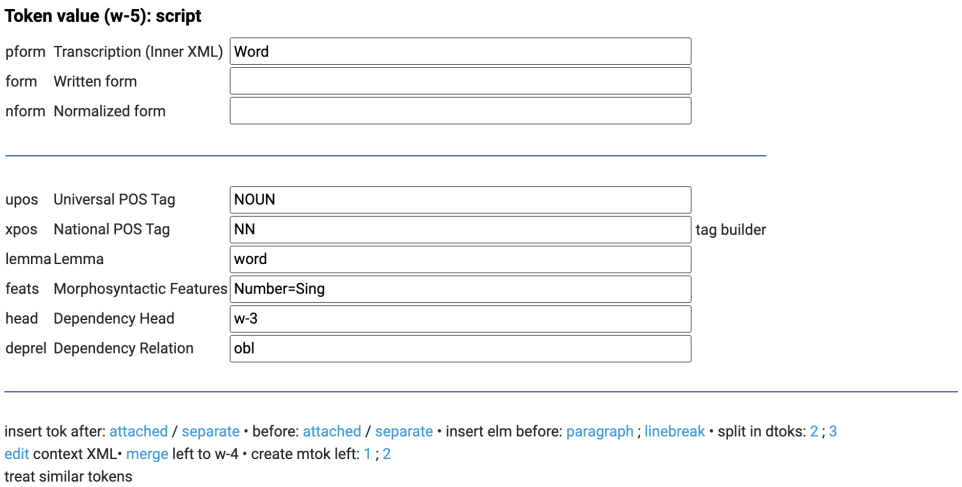

Figure 2: The Token Editor in TEITOK

So the sequence of ⟨tok⟩ nodes in the TEITOK document corresponds very directly with the sequence of lines in a CoNLL-U file. But all the other information in TEITOK is modeled quite differently. Generally speaking, a TEITOK/XML document can contain more information than a CoNLL-U file, and in a more structured fashion, for a number of reasons.

Firstly, there is no limit to the amount of attributes a (token) node can hold. And in TEITOK, you define for each corpus which attributes should be used. The example in figure **??** contains attributes relating to UD, but in a different corpus tokens can have very different attributes, some examples of which will be given in section 2.1.

[1] http://lindat.mff.cuni.cz/services/teitok/ud27/index.php

The fact that we can add attributes easily takes away the need to overload the `@misc` or other columns in UD with more and more information, since new types of information can get their own attributes. There are of course tabular formats where this is also possible, such as the CoNLL-U plus (conllup) format, where arbitrary numbers of columns can be specified. But for the representation of data with variable amounts of columns, XML tends to be a more amicable format. For instance, if we have attributes that appear only a handful of times in a large corpus, a tabular format like conllup forces you to add that attribute to all tokens, and be empty in the majority of them. In XML you can simply use the attribute only where needed. So new attributes can be added when necessary, without having to change anything in the existing files.

Secondly, everything in CoNLL-U that is not a token line is treated as a comment, with a largely loose convention of naming and content. Whereas in TEITOK, sentences and paragraphs are structural nodes, which just like tokens can be adorned with various types of annotations. This can be used for translation glosses as is found in CoNLL-U, but also for instance for phrasal typology, or for discourse relations defining dependencies between sentences rather than tokens. And structural elements are not limited to paragraphs and sentences - it is also possible to demarcate utterances, verse lines, chapters, and other types of segmentation of the text, each with their own set of attributes.

Thirdly, TEITOK can contain segments that should not be exported to the corpus. This is useful when the corpus contains parts that are not representative for the language. For instance, older or LRL text are not infrequently interspersed with text in another language - say Latin or English. And a typical format for LRL corpora is a non-native speaker interviewing a native speaker of the language. In both of those cases, the foreign or non-native parts should not be seen as representative of the language and hence should not be included in the treebank. But the content becomes uninterpretable by leaving those passages out. Therefore there are various ways in TEITOK to incorporate content that is not to be exported to the corpus, using for instance the ⟨`foreign`⟩ tag.

And finally, where CoNLL-U files typically contain little in the way to document metadata, in TEITOK each document is a separate TEI/XML file, with a full metadata header (teiHeader) that can specify almost anything about the text, such as for instance metadata about dialect, date, social status of the author, etc. And for many types of linguistic research, such metadata are often of crucial importance when working with corpus data. So when using a treebank for anything else than pure syntactic relations, it is vitally important to keep track of metadata. In general, TEITOK does not pose any restrictions on metadata, so the correct use in terms of representation, interpretations, and consistency is up to the corpus administrator. But for a specific corpus, or a collection of corpora such as in UDWiki, the obligatory metadata fields can be defined explicitly, and can be limited to fixed value lists to enhance consistency.

The idea behind TEITOK is not to modify the UD representation, which is why it uses the UD columns verbatim. But as shown above, the XML format offers the possibility to represent information differently, and makes it easier to incorporate new types of information as is often desired in treebanks.

2.1 Markup beyond UD

In TEITOK, the tokenization is not done over clean text, as is customary in traditional NLP, but rather over full TEI/XML documents, which can contain any type of TEI markup. This is often very important when working with anything but modern printed material: manuscript-based corpora will have added and deleted elements, unreadable and supplied element, and changes of hand, etc. Spoken corpora contain pauses, repetitions, truncations, etc. For the proper interpretation of the corpus, all those elements are often important, and flattening them out can lead to incorrect conclusions. There are UD treebanks that keep this type of information, such as for instance the spoken-specific data in the Slovenian Spoken Treebank (Dobrovoljc and Nivre, 2016), but those use largely ad-hoc solutions with limited expressive power. TEI offers a standardized representation for spoken phenomena (as well as many other types of (meta)linguistic information), which has been used, tested, and refined in a large number of projects, and offers a rich representation language.

The additional textual mark-up in TEI can also be used for stylistic information: headers, footers, bold, italics, small-caps, etc. All of these tend to have linguistic consequences, and when possible it is

much better to keep them. Keeping typesetting information makes it possible to read the entire document in a pleasant way. For modern printed English texts, that is not too important, but UDWiki is explicitly targeted towards LRL data - and for many of those languages, the treebank might be one of the few available resources online. By making the documents properly typeset, the treebank can serve as a collection of textual resources for the language community outside of the academic realm. By inserting ⟨tok⟩ nodes inside the existing TEI document, TEITOK keeps all this information, in a manner that does not interfere with the UD annotations over tokens, something that is hard to do in a tabular set-up.

As mentioned in the previous section, TEITOK documents can include any number of attributes, and not only fields to annotate dependency relations and morphosyntax. For instance, TEITOK can have (multiple) normalized orthographies for tokens. This is important in historical corpora, LRL corpora and chat-style corpora where there often is a lot of orthographic variation. Normalized orthography is kept explicitly as an attribute over the original content. Providing a normalized orthography enhances searchability. And both the original orthography and the normalized spelling(s) can be made searchable.

Additional attributes can also be used for semantic frames such as the ValLex frame in PDT, as is done in the TEITOK version of PDT-C[2]. And it has been used for several types of explicit codifications, such as an error code in learner texts to mark out deviations from the native norm, or linguistic codes to mark dialect-specific features in a text.

TEITOK documents can also have associated audio files, and furthermore have time-alignment between the utterances and the audio, making it possible to listen to the part of the audio corresponding to the utterance directly. And similarly, it can have facsimile image for historical corpora based on manuscript transcription, aligned not only with each page of the text, but also with each line or each word. Both of these options make it possible to directly verify the source material in case of potential transcription errors. An example of how all this information is exploited in TEITOK is given in figure 3, which shows an audio track from a video (shown on the bottom) with a transcription that scrolls the current utterance into view while playing the audio. This example was created from a subtitle (srt) file, and does not contain any speech-specific markup (see Janssen (2021) for examples with full speech markup). Crucially, this example is generated from a TEITOK/XML file that contains the full dependency parsing information in UD format.

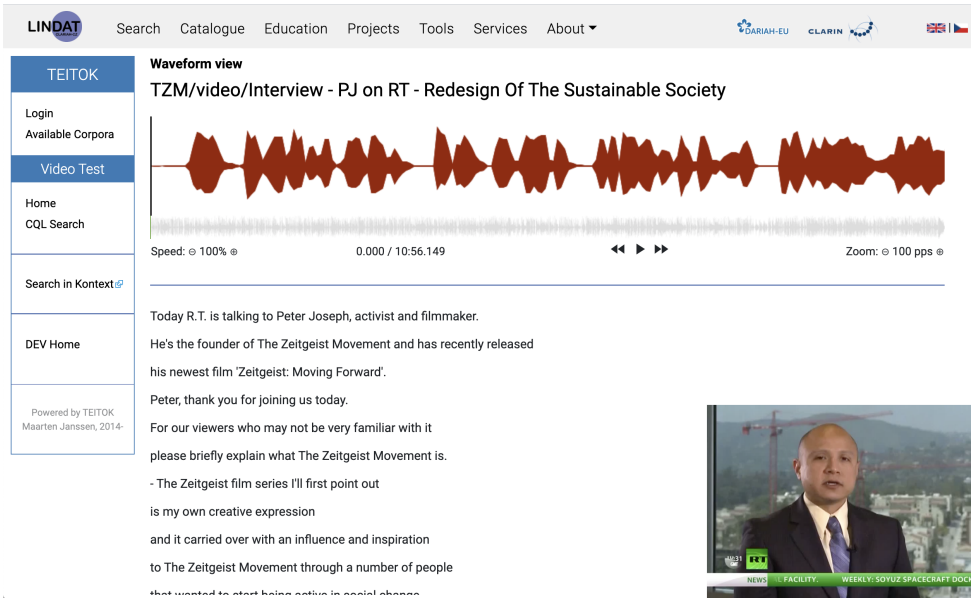

Figure 3: An example of an aligned video in TEITOK

TEITOK/XML files can also incorporate additional types of annotations, such as named entities with their respective classifications and linkings, quotations, references, footnotes, morphological decompo-

[2] https://lindat.mff.cuni.cz/services/teitok/pdtc/index.php

sitions, etc. So generally speaking, a treebank in TEITOK can be richer than what CoNLL-U supports. Of course, if TEITOK is used to create a treebank that is to be exported to the CoNLL-U format, much of this additional information will be lost in the export. But the TEITOK/XML files themselves do contain all the information, which can be exploited in a variety of different ways, while not hampering the representation of the core UD information. And the additional information is store in a structural way, so that if additional features are added to the UD format, it is just a matter of exporting that information in the appropriate way.

2.2 Searching

Searching in TEITOK is not done directly in the TEI/XML files, but rather by first loading the corpus files into some selected corpus query system (CQS). The CQS is used as an index over the corpus: a query is sent in the language of the CQS to the query system, which then returns a combination of the document ID and the token ID of the results. TEITOK then converts those results to XML fragments, and displays those fragments together with a link to the full original context. This means that all the information present in the original XML, including things that were not or cannot be exported to the CWB corpus, are present and visible in the search result. Since TEITOK has a modular design, the corpus can be made available via various CQS.

The default CQS in TEITOK is the Corpus WorkBench (CWB) (Evert and Hardie, 2011), which allows searches in the Corpus Query Langauge (CQL). When exporting to CWB, TEITOK also generates an index linking token IDs to byte-offsets in the original XML files. And with the method described above, the CQL results are rendered as XML fragments. The byte-offset index created for the CWB is also used to convert the results of any other CQS to XML fragments in an efficient manner.

CQL cannot (really) search in dependency trees, only in sequences of tokens, which is not ideal for UD treebanks. For the publication of UD2.7 in TEITOK, we therefore added the option at LINDAT to not only export the corpus to CWB (and Kontext[3], see Janssen (2021)), but also to PML-TQ (Pajas et al., 2009). PML-TQ is a query language meant explicitly to search dependency parsed corpora. And for the UDWiki project, a search module using Grew Match (Guillaume, 2019) has been added as well. The grew search works slightly differently, since Grew searches directly in CoNLL-U files. So each XML file in TEITOK is exported to a separate CoNLL-U file, with the token ID encoded in the `misc` column, which makes grew match results provide both the filename and the token ID (see also section 2.3).

SAll corpora in TEITOK can be updated upon request at any point, by running the indexing script from the interface. In that fashion, UD treebanks in TEITOK can be made searchable in all CQS after every correction or extension to the corpus. This means that the the corpus can become searchable directly, rather than having to wait for the next release of UD. For developers of the corpus this allows them to use the query language to search for possible errors in their corpus, by means of a range of query languages. And the result is linked directly to the XML file, which can be easily edited in TEITOK. For CQL, TEITOK even offers the option to edit directly from the KWIC results, which tends to be very helpful in known structural errors. For instance, the Spanish *que* can be either a relative pronoun, or a subordinating conjunction. And many taggers are surprisingly bad at distinguishing between the two. Such structural errors can be efficiently corrected by searching for constructions where we know the tagger gets it wrong, say the word *que* of which the head is a main verb that is not in the subjunctive, and then correcting the results - either by changing all of them in one go, or by manually checking them one by one.

2.3 Comparison

In order to get a good idea of the status of TEITOK as a tool for UD, it is important to compare it to other environments used to work with UD. On the one hand, there are dedicated tools to work with UD treebanks in CoNLL-U format, such as ConlluEditor (Heinecke, 2019) and Grew (Guillaume, 2019). And on the other hand there are generic tools that can be used for UD treebanks, the most popular

[3] `https://github.com/czcorpus/kontext`

amongst which are stand-off annotation tools like BRAT (Stenetorp et al., 2012) and ANNIS (Krause and Zeldes, 2016).

If we abstract away from the representation format, the interface of ConlluEditor is different from the tree editing mode in TEITOK, but the functionality is largely similar. However, the tree editor in TEITOK is only one module amongst many, and one that is open for improvement or even replacement if better option become available. It is just not possible to use UD tools like ConlluEditor directly in TEITOK due to the different storage format (although if so desired, such tools could be used by either adopting the tool to work with XML, or using an export-import strategy).

Grew is different in two respects. On the one hand, it does much more than just tree editing. It can include non-UD information such as sound alignment. And it can create and modify dependency trees by graph (re)writing. You could use grew parse in TEITOK for parsing, but there are no methods in TEITOK for automatically changing large amounts of data with grew rewrite rules or any other rules for that matter, and purposefully so: TEITOK is intended to be usable by people with limited computational skills. And when applied accidentally, large changes can render an entire corpus useless very easily. So all structural changes in TEITOK are always made from the command line, assuming people working from the command line are more aware of what they are doing, and will for instance make a backup of their data before any global changes.

And on the other hand, Grew match provides a search directly over the storage format (CoNLL-U) without any previous indexing, while in TEITOK it is always required to index the corpus after making (large) changes, making for a sometimes cumbersome intermediate step. But (almost) all CQS use indexing, and do so to increase search speed. CWB itself natively indexes over VRT files. Tools built upon database systems, like PML-TQ using PostGres, use the native indexing of the database system (and often load the data from a different format to start with). And even tools working directly with XML files such as BlackLab[4] or corpora built in ExistDB[5] first create an index on way or another. Grew match successfully demonstrates that for existing UD treebanks, on-the-fly indexing works sufficiently fast. But UD is starting to get used in larger corpora, the largest at the moment probably being InterCorp[6], with in total over 1G tokens. And for such large corpora, on-the-fly indexing is unlikely to be fast enough. So in the long run, indexing is likely always to be a necessary step.

The stand-off tools are based on the idea that you should keep the original data unmodified, and keep all annotation in fully independent files. That means a fundamentally different type of representation than used in CoNLL-U[7]. Stand-off annotation has both advantages and disadvantages which go well beyond the purpose of this paper. But although tools based on stand-off annotation like BRAT and ANNIS can contain all the information required for dependency trees, and as such can be used to create UD treebanks, they do so in a fundamentally different manner. The fact that TEITOK uses a format that is very close to CoNLL-U makes it closer to a native UD tool than a stand-off solution can be.

3 UDWiki

UDWiki is a TEITOK meta-project, which is to say a project that defines common settings for all projects under its umbrella. It predefines all the settings needed to create a UD treebank, making all UDWiki use the same set-up to increase consistency. It also contains a collection of specific modules that are custom build to help in the process of creating, annotating, and publishing UD treebanks. The annotation process will be described in the next section. The system is currently fully operational, yet still in beta, and can be found at: `https://quest.ms.mff.cuni.cz/teitok-dev/teitok/udwiki/index.php`

The idea behind the workflow of UDWiki is as follows: users can ask for the creation of a new UDWiki treebank. We will then create a TEITOK project within the UDWiki meta-project, as well as a UD git repository. The user is then asked to fill in several forms about the project, with the required metadata,

[4] `http://inl.github.io/BlackLab/`

[5] `http://exist-db.org/`

[6] `https://intercorp.korpus.cz/`

[7] Although the more recent options to keep character ranges in UD (TokenRange) move more in this direction

and in the case of a new language also the data needed for the language description. The user can also add additional users in the case of collaborative projects.

Once set-up, the users can start to fill their corpus with documents, and start annotating them. And once there are annotated corpus texts, the whole corpus can be pushed to the UD git repository, exporting the TEI/XML files to CoNLL-U, dividing them into development, test, and train sections, and compiling the additional files such as the README, the LICENCE and the CONTRIBUTING.

UDWiki was built upon CorpusWiki, and our experience in that project is that many people are not content building their own work on the server of someone else. That is likely to be less of an issue for UDWiki since the UDWiki data are always available via the git (although there are always cases where people want to keep the data private, for instance for PhD projects, where students often do not want their data to be public until the defense). That is why we make it explicitly possible for people to set-up their corpus in UDWiki, but download the entire project at any point if they choose to rather continue the work on a local TEITOK installation. And we also intend to keep the UDWiki system itself open source so that people can run it locally if so desired, although the amount of tools used in the background will likely make it not easy to set-up a full local version of UDWiki.

As an environment for UD treebanks, UDWiki attempts to provide an easy interface to use as many of the UD tools as possible. Where possible, those tools will be rewritten to work directly with the TEI/XML format, while for other tools the TEITOK data are first exported to CoNLL-U, after which the tool is applied. A good example of a CoNLL-U based tool is the official UD validation tool. Since the validator is an official tool, and frequently updated, it is not a good idea to make any modifications to the tool. Rather, the tool is used directly from a Git clone of the official tool repository[8] over an export of the XML file to CoNLL-U. An example of the output of the validator in UDWiki is given in figure 4, taken from the tentative treebank for Papiamento (see section 4.2). The first line specifies the name of the XML file we are editing (nanzi_kriki.xml). The third line states that there is no language definition for Papiamento in UD yet - with UDWiki rather working with a local definition that should be moved to the UD repository once sufficiently established. And below that is the raw output of the validator script. To make it easy to correct the errors encountered by the script, the pointers to the .conllu file have been replaced by hyperlinks to the original XML: so the `Line 6` in the first error line links to the editor for the corresponding token, which has the ID w-8 in nanzi_kriki.xml. And the `s-1` after it links to the tree editor for the first sentence in that same file (shown in figure 5).
UDWiki is intended as an open-ended system. Which extensions will be added will depend on the needs of the community. For instance, we have been told that some groups working on UD treebanks would want to see their treebanks to be searchable while they are working on them, and not having to wait until the next release of UD. Mostly such teams will have their own workflow and might not want to switch to using TEITOK. For those cases, it will be possible to set-up a dummy UDWiki project that is not maintained in UDWiki, but for which the XML files are pulled from the Git repository, converted to TEI automatically and then made searchable in the various CQS.

3.1 Annotating

TEITOK has a built-in tree editor for dependency trees. The editor draws the tree, and then lets you either reattach nodes or change their dependency labels. The interface is shown in figure 5 which shows the name of the file, the sentence itself (as an XML fragment), and any sentence metadata below that, such as an English gloss in this case. By clicking on a node, you can reattach it to any other node. By clicking on a dependency label (as in the example on the *amod* label) you can select a new dependency label from a drop-down list. And by clicking on the word in the sentence, you can edit token annotations as shown in figure 2. You can also edit the metadata for the sentence which will pop up an HTML form.

For languages for which there already is a UD treebank, it is possible to run UDPIPE (Straka and Straková, 2016) to automatically parse the files in the corpus using a simple click. That will export the corpus to CoNLL-U, run it through the UDPIPE REST service, and load the results back into the TEI/XML file. Once parsed, the sentences can then optionally be manual verified. When manually

[8] `https://github.com/universaldependencies/tools`

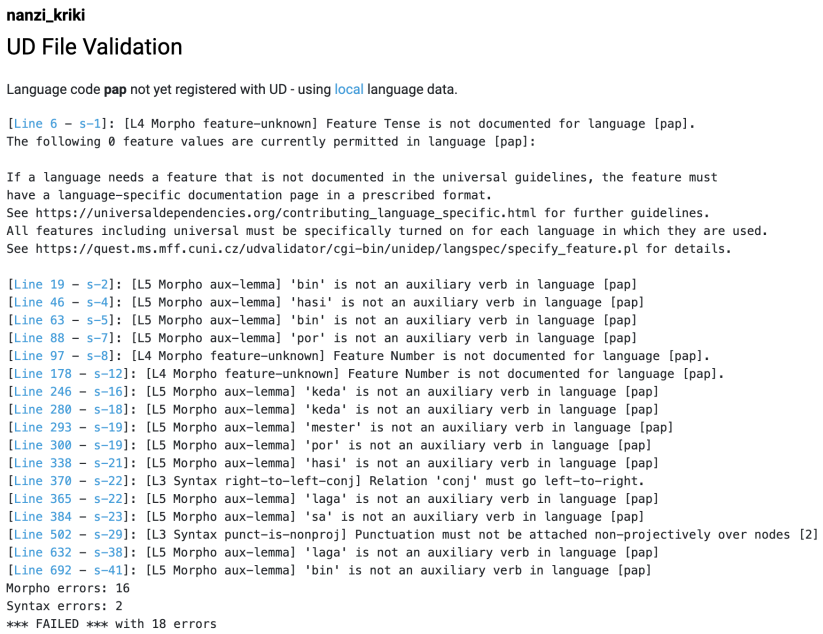

Figure 4: The UDWiki interface for the validator

correcting trees, UDWiki keeps track of the status of each sentences, which can be explicitly set to correct or incorrect, to get an overview of the progress.

But UDWiki is explicitly set-up to help people to build treebanks for new languages. And for those there will be no models in UDPIPE (or typically any other UD parsers). For such corpora, UDWiki offers the functionally inherited from the CorpusWiki project: to use an incrementally built POS tagger and/or parser. In this way, TEITOK can be used to manually annotate corpora with UD.

3.1.1 Incremental Tagging and Parsing

The way incremental tagging in CorpusWiki worked is that the corpus is started with a single file, which is manually annotated. Since manual annotation is slow and labour-intensive, the recommendation was to start with a short file, in the range of 500-1000 tokens, ideally a translation/retelling of a popular tale. Once the initial file was fully annotated, the interface provided the option to train the NeoTag POS tagger (Janssen, 2012) on that single file, and use the resulting parameters to automatically tag the second file in the corpus. The accuracy of that tagger will naturally be low, in our experience in CorpusWiki, typically somewhere around 40% for an initial training file of 800 tokens. Yet even with low accuracy, this means that the second file is easier to annotate than the first. And by retraining the tagger after each new file that has been corrected, the accuracy rises quickly, typically reaching 90% accuracy after the first 5000 tokens.

For UDWiki, a similar set-up has been created around the UDPIPE parser. For each text in the treebank, the annotation status is kept in the header: whether it is only tokenized, POS tagged (and ideally verified), or parsed. A CoNLL-U file is created for all the tagged files, and a separate one for all the parsed files. From those files, a tagger model and a parser model are trained separately, since especially initially, there might be many more text that have been tagged than parsed. And the models can then be used to tag and parse the next file, correct, and retrain. This makes it possible to have a more accurate tagger and parser with each new file as in the Corpuswiki set-up.

The choice of UDPipe for the parsing is because it is a convenient and well-established tool that is furthermore developed within the same institute as UDWiki. But if other tools prove to be more efficient for the process, it is easy enough to modify the workflow to work with another parser. The parser just has to be trainable automatically from CoNLL-U files without manual intervention, and be sufficiently fast to allow rapid subsequent training sessions.

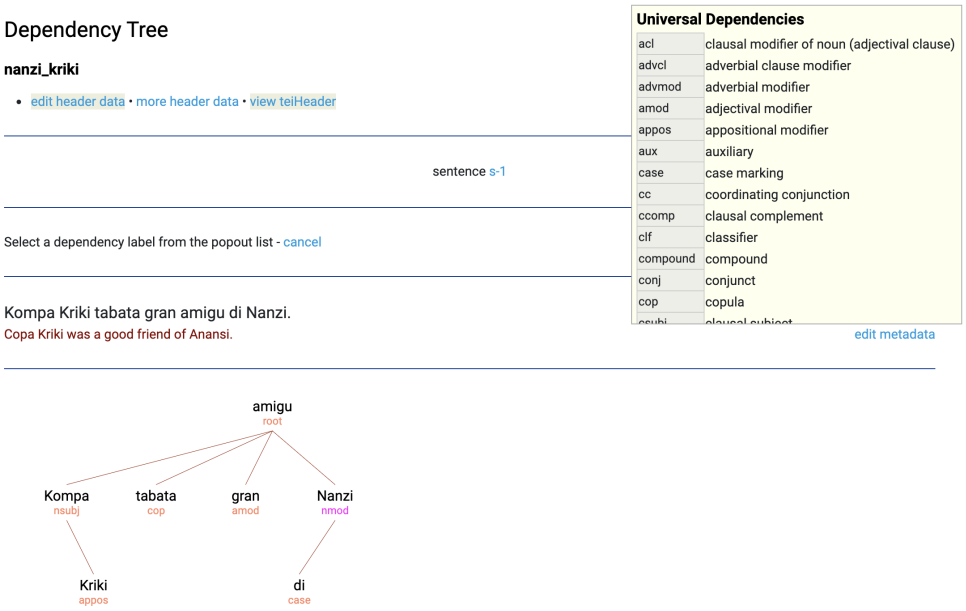

Figure 5: The UDWiki Tree Editor

3.1.2 Guided Tagging and Parsing

Starting to build a treebank is not easy - there are a lot of things to get used to. People not used to POS tagging will have to get used to cases where it is not straightforward to tell the POS of a given example; UD has interpretations of features that do not always coincide with traditional grammar books; and the logic of the direction of a dependency relation can go contrary to the order in a constituency tree. The idea behind UDWiki is to provide as much interactive feedback to the user as possible to overcome those issues: to provide explanations about features and their values, but also to show which tags were already used for a given word in the treebank - or in other treebanks for the same language, or even in treebanks for similar languages. And not only for morphosyntax, but also for dependency relations: which is the most typical dependency relation for a given type of word, with examples. And it can fill in the same features for identical words without having to copy them to each word, which especially for the first document makes the process a lot faster.

The standard TEITOK interface can already provide many of those functions. But we are working on several interface modules specific for UD that makes this process more streamlined, and speed up the annotation process. For new languages, there is also an editor to set up the language definitions (initially in local files): to define the auxiliary verbs, the features and their values. The verification tool in figure 4 helps to pinpoint any errors in the current XML file. And when editing dependency trees, we are setting up on-the-fly verification so that the system warns immediately about any deviations from the UD norms in the current tree.

Like with the rest of the design of UDWiki, exactly which methods are most effective is something that will only become apparent by the system being used, to find out where the largest bottlenecks are, and then attempt various strategies to resolve those bottlenecks.

4 Use Cases

Given that UDWiki is not yet really released, the number of use cases is still limited to some test cases and two treebanks that are still in their initial stages. But TEITOK, and its predecessor CorpusWiki, have been used for a wide range of corpora, mostly restricted to (position-based) POS tagged corpora. The system has been very positively received, with people with little to no computational background able to build their own annotated corpora. This section will review some relevant examples.

4.1 POS Tagged Corpora

There are various LRL corpora that have been built in TEITOK, including CoDiaJe[9] (Ladino), EModSar[10] (Sardinian), and LUDVIC[11] (Caboverdean). None of these corpora are using UD, nor are they annotating dependencies. But they do serve as examples of how annotated corpora can be created for new languages in TEITOK. All three LRL corpora listed above use a locally trained NeoTag tagger, using folders to indicate which files should be used as a training corpus. For POS tagged corpora, they are all of modest size, but they are of a significant size for a treebank. EModSar is the smallest, with 6K tokens, LUDVIC is considerably larger with 100K tokens, and CoDiAJe is the largest with 800K tokens. All three are still being developed further. And with a minimal amount of initial support, all three projects have been completely independent in building and annotating the corpus, using the incremental tagging described in section 3.1.1.

Of specific interest in these corpora is CoDiAJe, since it has a feature that is difficult to deal with in plain CoNLL-U: Ladino over time has been written in various different writing systems (Latin, Hebrew, and even some Cyrillic). Therefore, CoDiAJe is a mixed corpus in which not all documents are written in the same way. Nevertheless, since all tokens are provided with a normalized (romanized) orthography, it is still a homogeneous corpus. And since search results in TEITOK are rendered as XML fragments, the results (by default) show in their original writing system.

Another example of a POS tagged corpus built in TEITOK is OLDES (Janssen et al., 2017), a corpus of Old Spanish, with around 20M tokens. OLDES was not annotated from scratch with NeoTag, but rather annotated automatically with Freeling (Carreras et al., 2004), then improved manually in TEITOK, after which a NeoTag parameter set was trained on the corpus. OLDES made use of the various types of efficient editing options provided by TEITOK to improve the automatically assigned tags.

4.2 Treebanks

The only two treebanks thus-far that are aimed at becoming full UD treebanks are the spoken Occitan corpus (UD_OCI-OCOR), and a thusfar small Papiamento corpus (UD_PAP-UFAL), both available via the UDWiki website.

OCI-OCOR is not a new corpus built in UDWiki, but rather a conversion of an existing corpus: OCOR, a corpus of Occitan oral narratives (Carruthers and Vergez-Couret, 2018). The original files of OCOR were downloaded from the Zenodo repository[12], and added to a UDWiki project with minimal modification since the files are already in the TEI format. The data were then parsed using the Talismane parameters created for the UD Occitan treebank (Miletic et al., 2020). And finally, the parsed data are being manually corrected to get a gold standard treebank for spoken Occitan. Since there already was a UD parser for Occitan, OCI-OCOR does not really show the full intent of UDWiki. But it does show that UDWiki is easy to use for languages for which there already is a parser. And it helped to provide easy access to a valuable corpus that was thus-far hard to use and not searchable online.

PAP-UFAL is a thus-far small treebank built from several texts in Papiamento collected from the web, containing at the moment two annotated texts of 1700 tokens in total. The treebank has been built from scratch in TEITOK, starting from HTML documents, and tagging and parsing them directly in the tool. It is a proof-of-concept treebank for a language for which very few (NLP) resources exist. To make the treebank accessible for non-speaker of the language, both sentences and words are glossed with English translations. The first file in the Papiamento treebank was annotated manually, and the second file was tagged automatically with the UDPipe model trained on the first text. The results are as good as one might expect.

[9] http://corptedig-glif.upf.edu/teitok/codiaje/
[10] http://corpora.unica.it/TEITOK/emodsar/index.php
[11] http://teitok.clul.ul.pt/ludvic/
[12] https://zenodo.org/record/1451753\#.YUxI1p4zblw

5 Conclusion

As shown in this paper, UDWiki is a complete environment for building UD treebanks from scratch, especially for LRL, without requiring much computational knowledge from the users. This will hopefully attract linguists interested in creating treebanks for new languages, which otherwise would not have managed to create one, or allow native speakers of languages to help out in the creation and extension of existing treebanks.

UDWiki was conceived as way to use TEITOK to generate normal UD treebanks, to be exported as CoNLL-U files and included in the UD infrastructure. But as shown, TEITOK files themselves can contain more information than CoNLL-U allows, information that has to be either removed or flattened down when exporting to CoNLL-U. So TEITOK can also be used as a richer storage format for UD treebanks, where the dependencies and morphosyntactic information is stored according to the UD standards, while all other information is stored in whichever format is most appropriate for it, whether it be metadata, spoken annotations, time mark-up, or anything else. To do that consistently, it would of course be necessary to establish annotation standards and add consistency check for everything considered core content of the extended treebanks.

An initiative like UDWiki only works if it is being used - only with hands-on experience and feedback can the system be fine-tuned to work in an optimal way. There are various questions that have not been established, such as how to use the Git repository: whether pushing the data to the repository should be done constantly, with a pre-determined frequency (say daily) or upon user request. Whether the UDWiki repositories should be kept separate from the core UD repositories (and then synced) or not. And whether only the CoNLL-U files should be exported, or also the TEITOK/XML files. And as mentioned before, the functionality of the interface is kept purposefully open in order to be able to demands coming from the community. And those needs could be of various types: changes to the interface, using a better parser, changing the workflow, adding more verification and helping tool, or adding tools for areas that are not part of UD but very helpful for LRL (or other) corpora.

An example of an area that might have to be added to UDWiki is Interlinear Glossed Texts (IGT): the experience with working with small languages has shown that without any description of their morphology, it is often impossible to start tagging immediately. The first step (field work) is to collect texts and establish the morphology, which is typically done in IGT. TEITOK has support for IGT, so in order to allow people to directly build treebanks for such languages, it might be necessary in UDWiki to let people initially create an IGT corpus, and then guide them in converting that IGT corpus to UD, not by removing the IGT data but rather by adding the UD attributes to existing IGT corpus in TEITOK.

And finally in TEITOK, we are gradually moving to make UDPipe the default tagger (and parser). There are many different types of corpora in TEITOK: historical, learner, dialect, spoken, with often additional annotation layers such error annotation, named entity tagging, rhyme schemes, etc. By having UDPipe as the default tagger, there is hence effectively a growing number of UD treebanks of different types, for document types where that would not easily happen if the treebanks would have to be written in CoNLL-U, since the additional annotations required could not (easily) be incorporated.

References

Xavier Carreras, Isaac Chao, Lluís Padró, and Muntsa Padró. 2004. Freeling: An open-source suite of language analyzers. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*.

Janice Carruthers and Marianne Vergez-Couret. 2018. Méthodologie pour la constitution d'un corpus comparatif de narration orale en Occitan : objectifs, défis, solutions. *Corpus*.

Kaja Dobrovoljc and Joakim Nivre. 2016. The Universal Dependencies treebank of spoken Slovenian. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1566–1573, Portorož, Slovenia, May. European Language Resources Association (ELRA).

Kaja Dobrovoljc, Tomaž Erjavec, and Nikola Ljubešić. 2019. Improving UD processing via satellite resources for morphology. In *Proceedings of the Third Workshop on Universal Dependencies (UDW, SyntaxFest 2019)*, pages 24–34, Paris, France, August. Association for Computational Linguistics.

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

Bruno Guillaume. 2019. Graph Matching for Corpora Exploration. In *JLC 2019 - 10èmes Journées Internationales de la Linguistique de corpus*, Grenoble, France, November.

Johannes Heinecke. 2019. ConlluEditor: a fully graphical editor for Universal dependencies treebank files. In *Universal Dependencies Workshop 2019*, Paris.

Maarten Janssen, Josep Ausensi, and Josep Fontana. 2017. Improving POS tagging in Old Spanish using TEITOK. In *Proceedings of the NoDaLiDa 2017 Workshop on Processing Historical Language*, pages 2–6, Gothenburg, May. Linköping University Electronic Press.

Maarten Janssen. 2012. Neotag: a pos tagger for grammatical neologism detection. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *LREC*, pages 2118–2124. European Language Resources Association (ELRA).

Maarten Janssen. 2016a. Pos tagging and less resources languages individuated features in corpuswiki. In Zygmunt Vetulani, Hans Uszkoreit, and Marek Kubis, editors, *Human Language Technology. Challenges for Computer Science and Linguistics*, pages 411–419, Cham. Springer International Publishing.

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

Maarten Janssen. 2018. TEITOK as a tool for dependency grammar. *Procesamiento del Lenguaje Natural*, 61:185–188.

Maarten Janssen. 2021. A corpus with wavesurfer and tei: Speech and video in teitok. In Kamil Ekštein, František Pártl, and Miloslav Konopík, editors, *Text, Speech, and Dialogue*, pages 261–268, Cham. Springer International Publishing.

Thomas Krause and Amir Zeldes. 2016. Annis3: A new architecture for generic corpus query and visualization. *Digital Scholarship in the Humanities*, 31(1):118–139.

Aleksandra Miletic, Myriam Bras, Marianne Vergez-Couret, Louise Esher, Clamença Poujade, and Jean Sibille. 2020. Building a Universal Dependencies treebank for Occitan. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2932–2939, Marseille, France, May. European Language Resources Association.

Petr Pajas, Jan Štěpánek, and Michal Sedlák. 2009. PML tree query. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

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 *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, page 102–107.

Milan Straka and Jana Straková. 2016. UDPipe. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.