The CLARIN-DK Text Tonsorium

Bart Jongejan
Department of Nordic Studies and Linguistics
University of Copenhagen, Denmark
bartj@hum.ku.dk

Abstract

The Text Tonsorium (TT) is a workflow management system (WMS) for Natural Language Processing (NLP). The software implements a design goal that sets it apart from other WMSes: it operates without manually composed workflow designs. The TT invites tool providers to register and integrate their tools, without having to think about the workflow designs that new tools can become part of. Both input and output of new tools are specified by expressing language, file format, type of content, etc. in terms of an ontology. Likewise, users of the TT define their goal in terms of this ontology and let the TT compute the workflow designs that fulfill that goal. When the user has chosen one of the proposed workflow designs, the TT enacts it with the user's input. This untraditional approach to workflows requires some familiarization. In principle, the TT cannot predict which of the proposed workflow designs is most appropriate, because the text may have peculiarities that are as yet uncharted. The user has to make the choice. In this paper, we reflect on the experiences with providing, testing and using workflows aimed at annotating transcripts of parliamentary debates. We propose possible improvements of the TT that can facilitate its use by the wider CLARIN community.

1 Introduction

Many developments in workflow management systems (WMS) are aimed at bringing down workflow execution time and handling ever bigger amounts of data. In the user community that CLARIN addresses, on the other hand, ease of use, especially for users with a nontechnical background, and adaptability to special needs, are often more important than speed and data size.

Our aim is to let small and medium scale scholarly projects benefit from an easy to use and open WMS that manages a well-maintained collection of state of the art NLP tools. This WMS is the Text Tonsorium (TT). It was constructed by the CLARIN-DK staff (Offersgaard et al., 2011), but it has been away from the clarin.dk web site for some years. After many technical improvements, it is again part of CLARIN-DK, this time with a new interface.

Traditionally, workflow management systems require (expert) users for the construction of workflow designs. A good example of such a system is WebLicht[1] (Hinrichs et al., 2010). The characteristic that sets the TT apart from traditional WMSes is that workflow designs are computed automatically. Central to this computation is an ontology that is used to describe all data objects involved in workflows. Tools, too, are described by this ontology, since the TT, when computing workflow designs, only needs to know what kind of data enters a tool and what kind of data comes out. A technical description that explains how the TT computes workflow designs is in Jongejan (2016). More about the user perspective of the TT, as conceived during the DK-CLARIN project, is in Offersgaard et al. (2011) and in Jongejan (2013).

Users experience the TT as a very different service, compared to traditional workflow management systems, because the TT dispenses with the expert user who creates and shares workflow

[1] https://weblicht.sfs.uni-tuebingen.de/weblichtwiki/index.php/Main_Page

Bart Jongejan 2021. The CLARIN-DK Text Tonsorium. *Selected papers from the CLARIN Annual Conference 2020*. Linköping Electronic Conference Proceedings 180: 180 111–121.

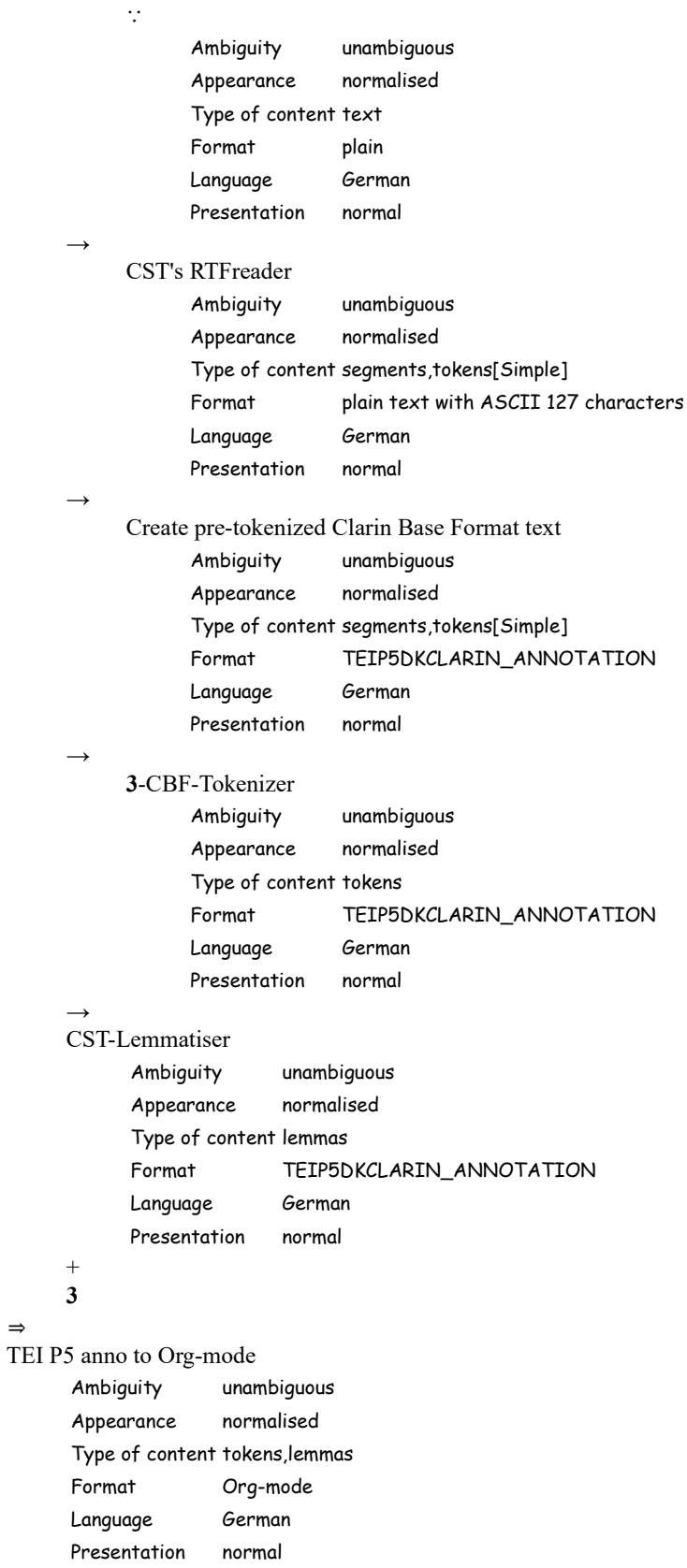


Figure 1: Graph of a workflow design with five tools. The flow is from top to bottom. The output from tools marked with a number is sent to the next tool and to any tool that has the same number mentioned as input. Here, the 3rd tool sends output to the 4th and the 5th tool.

designs with other less expert users. True, users do not have to think about the technisch aspects of workflows, but users must decide for themselves which of the automatically computed workflow designs to choose. This choice can be difficult and time consuming, and would have been made once and for all, if a human expert user had created a workflow design for them. It is therefore necessary to offer guidance to inexperienced users. On the one hand, metadata about tools and datatypes must be presented in an understandable way and in the places where they can be helpful. On the other hand, knowledge and experiences must be shared between users, first and foremost by way of named, descriptive bookmarks that can be attached to workflow designs that have been tried out and used.

The structure of this paper is as follows. Section 2 is about the current state of the TT. Section 3 presents a condensed technical outline of the TT. Section 4 explains how the data is described by metadata and how the tools are described in terms of the metadata characterizing their input and output data. Section 5 is a case study of the design, implementation and use of the Danish ParlaMint workflow design. In Section 6 we outline an improvement that some users have proposed, but that cannot be realized. To make good for this, Section 7 is an exposition of our plans for making an interface to the TT that can be easy to use by the larger CLARIN community. Section 8 tells where the TT can be found on the internet. Finally, in Section 9, concluding remarks follow.

2 Current State

There are currently over 40 tools[2] integrated in the TT, spanning from tokenization to syntactic parsing and from PDF-to-text conversion to text-to-speech transformation. Some tools were developed to address the needs of a single project and later generalized to make them useful for a wider segment of researchers. An example of spin-off from two consecutive and unrelated projects is that the TT is able to annotate a wide class of TEI P5 formatted texts with lemmas, part-of-speech (PoS) tags and syntactic dependencies.

Many of the tools are multi-lingual. The CST lemmatizer (Jongejan and Dalianis, 2009) (CSTlemma), for example, lemmatizes 28 languages[3]. The Danish linguistic resources for CSTlemma cover three historical periods: medieval, late modern and contemporary. A few tools, such as the Named Entity Recognizer, work only for Danish.

3 Technical Outline

3.1 Software Components

The TT is a web application that consists of a hub and a webservice for each tool that is orchestrated by the TT. The integrated tools communicate with the hub by the HTTP protocol, but do not communicate with each other. Users interact only with the hub.

3.2 Tool Integration

The TT offers an easy way of embedding a tool in an ecosystem of already existing tools. First, a tool provider visits the administration page of the TT and enters boiler plate metadata (ToolID, ContactEmail, Version, Title, ServiceURL, Publisher, ContentProvider, Creator, InfoAbout, Description) as well as metadata that describes the input and the output of the tool in terms of language, file format, and a few other dimensions. The TT then creates a program stub in the PHP language for a web service that is already tailored to the tool to be integrated.

3.3 Workflow Composition

The TT uses dynamic programming with memoization to compute all workflow designs that combine tools such that the output will be in agreement with the user's specifications, given the

[2]See `https://cst.dk/texton/help`

[3]Some of training data sets with which CSTlemma was trained, to wit the MULTEXT East free and non-commercial lexicons (Erjavec et al., 2010a; Erjavec et al., 2010b), were found in the Slovenian CLARIN portal.

user's input. Computation of these designs, pruning unlikely designs and removing irrelevant details from the presentation of the list of remaining candidates, is relatively fast, given that there may be thousands of viable designs to sift through. This process can take anywhere from a few seconds to a couple of minutes.

Fig. 1 shows the full details of a single workflow design. In this case, the TT had recognized the input as plain text and the user had defined the goal as German lemmas. The shown workflow design is one out of 50 designs, a number that would have been reduced to ten if the user also had mentioned that the output had to be in ORG-mode format.

3.4 Workflow Enactment

When the user has chosen one of the proposed candidate workflow designs, the TT enacts that workflow with the data that the user has uploaded as input. The input, intermediary results and final results are temporarily stored on the computer on which the TT runs.

The results from each step in a running workflow can be inspected as soon as the step has been executed. The results can be downloaded as a zipped archive. The user can choose whether or not to include all input and intermediary results in the zip-file.

3.5 Data Formats

The TT handles Office documents, TEI P5 documents, HTML, PDF, images of text, plain text, JSON, ORG-mode tables, CONLL (14 column CONLL-2009 as well as 10 column CoNLL-U), CWB (Corpus Workbench), bracketed (Lisp-like) text, and WAV sound files. Some formats are only available on the input or on the output side.

The TT is primarily designed for annotation tools that output their result without also copying the input to the output. By allocating stand-off annotations in files separate from the input and from other annotations, the TT has the freedom to send intermediary results from earlier processes in any combination as inputs to later processes, thus circumventing a need to have data definitions for each possible combination.

Since users normally require output that contains results from several workflow steps combined, some data definitions for combinations of text and/or annotations have been made, employing expressive formats such as JSON, ORG mode tables, CONNL, CWB and TEI P5. Fig. 1 illustrates this: tokens and lemmas are in separate files, the first as the output of a tokenizer and the latter as the output of a lemmatizer. These two intermediary results are both sent to the final tool, which combines the tokens and the lemmas in a table with two columns, using the ORG-mode formatting mode.

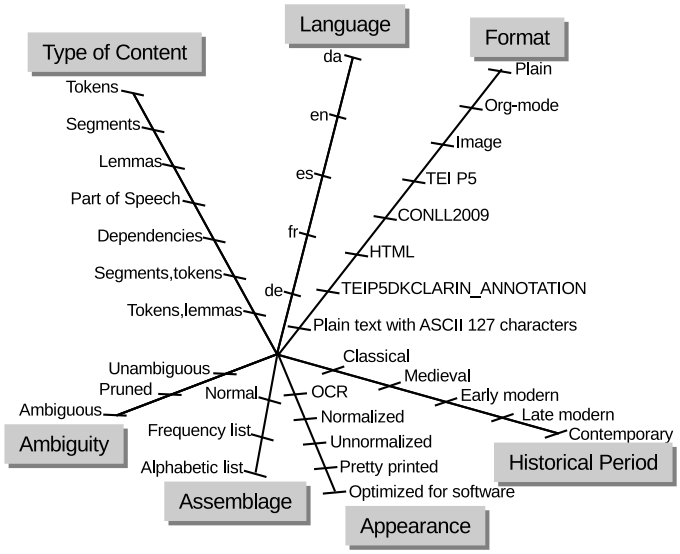
Complex data types are not restricted to the final result of a workflow. If there is a tool that accepts a complex data type, workflow designs can be computed that take user input of that type. Such data types can also occur as intermediary result.

4 Metadata

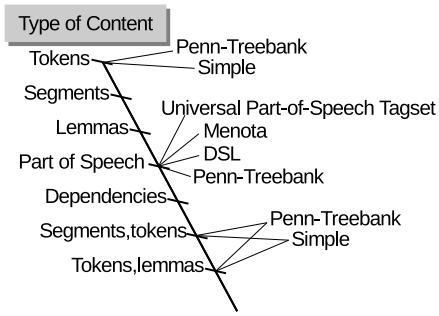
Seven dimensions are used to describe data: the *Type of Content*, the *Language*, the *Format*, the *Historical Period*, the *Appearance*, the *Assemblage*, and the *Ambiguity*, see Fig 2.

The TT treats all dimensions on an equal footing: it does not care whether a tool transforms data between two *Languages* , or between two *Formats*, or between two *Types of Content*.

Each of these dimensions needs to be populated with values. For example, this is the current list of values that *Type of Content* can take: **text**, **tokens**, **sentences**, **segments**, **paragraphs**, **PoS tags**, **lemmas**, **word classes**, **(syntactic) dependencies**, **tagged terms**, **named entities**, **morphemes**, **noun phrases**, **repeated phrases**, **N-gram frequencies**, **keywords**, **multiple word terms**, **lexicon**, and **head movements**. These *Types of Content* are primitive. There are also some complex *Types of Content* that combine two or more primitive *Types of Content*. For an example of their use, see Section 3.5 and Fig. 1, where we see two examples of complex *Types of Content* that combine more primitive *Types of Content*: **segments,tokens** and **tokens,lemmas**.



(a) Example of independent dimensions.



(b) Examples of subspecification of some values in the *Type of Content* dimension.

Figure 2: Dimensions and subspecifications of values along one of the dimensions.

The TT handles one more level of specification. In this extra level it is possible to discern variants of a given value in a dimension. For example, there may be a need to discriminate between Universal PoS tags and the PoS tag set used in the Penn Treebank, which are specifications of the value **PoS tags** in the *Type of Content* dimension. Similarly, it might be useful to distinguish between JPEG, PNG and SVG, which are specifications of the value **image** in the *Format* dimension. See Fig 2. The purpose of this extra level of information is to make matching tools with each other, with input data and with output requirements, more forgiving. It also helps to keep technical details away from the user.

5 Use case: Annotation of a Danish Parliamentary Corpus

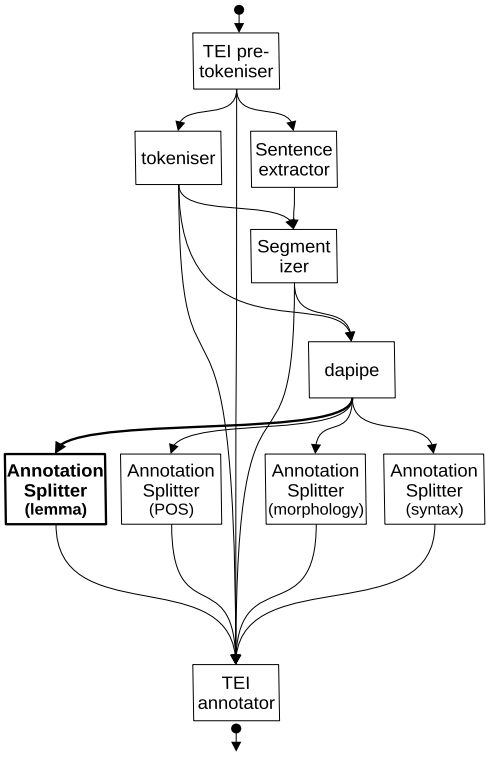
The ParlaMint project[4] has the aim to make national parliamentary data in several countries available in a uniform format and enriched with linguistic annotations. The chosen format follows the TEI P5 guidelines. We have decided to annotate the Danish parliamentary data with morpho-syntactic descriptions (**msd**), lemmas and syntactic dependency relations.

For handling the linguistic annotation process of the Danish ParlaMint data, the choice fell upon the TT, since it already had several tools for transformation and annotation of TEI P5 data.

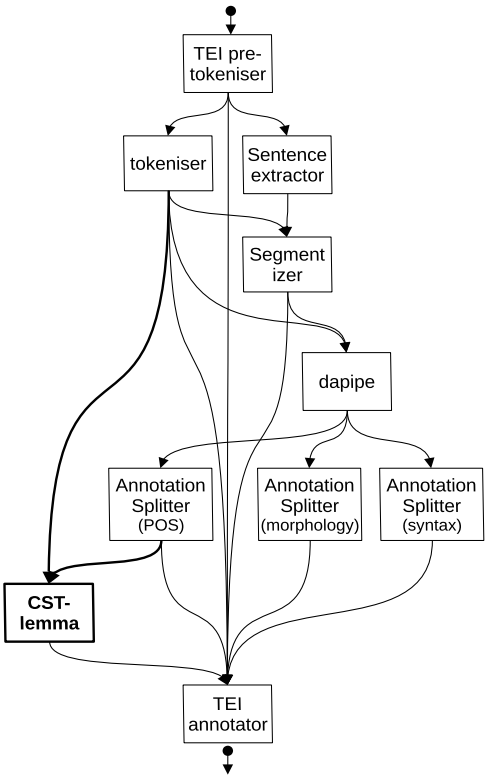
A tool for annotation of Danish (plain) text with **PoS tags**, **lemmas**, and syntactic **dependency** relations was already integrated in the TT. This tool, dapipe[5], is based on UDpipe (Straka and Straková, 2017) and trained by colleagues at the IT University of Copenhagen. Per default, dapipe expects plain text and does everything necessary: segmentation, tokenization, PoS tagging, morphological analysis, lemmatization, and syntax analysis. Our first experiment was to iterate over all `<seg>` elements in each input file, for each element (1) extracting the content, (2) saving it in a plain text file, (3) running dapipe with that file as input, and (4) transform dapipe's output to follow the TEI P5 standard. This was the most straightforward way to use dapipe in the ParlaMint project, but it caused a considerable overhead, since dapipe had to be started and initialized with the Danish language model for every single utterance. Then we discovered that there is a possibility to feed dapipe with an already tokenized and seg-

[4]ParlaMint: Towards Comparable Parliamentary Corpora (`https://www.clarin.eu/content/parlamint-towards-comparable-parliamentary-corpora`)

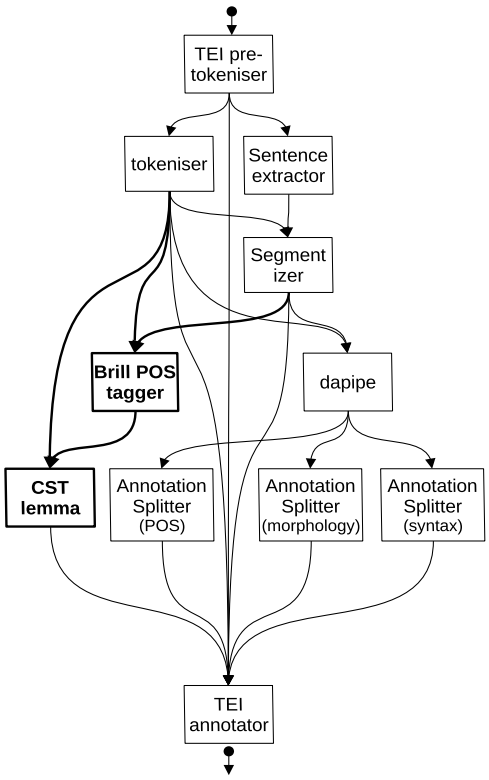
[5]`https://github.com/ITUnlp/dapipe`



(a) Workflow using dapipe for all linguistic annotations.



(b) Workflow using CSTlemma for better lemmatization. CSTlemma using dapipe's POS as hint.



(c) Workflow that uses an alternative tagger to provide hints to CSTlemma.

Figure 3: Three candidate workflow designs (out of 21) that were tried for annotating transcripts of parliamentary debates in Denmark. (b) was chosen because it was the only design guaranteeing congruence between lemmas and PoS tags.

mentized input. We decided to choose another, more complex but more efficient solution: first tokenize and segmentize the input, using TEI elements to indicate the **tokens** and **segments**, and then send these preprocessed data through dapipe in one go. Dapipe's output could then be matched to the proper **tokens** and **segments**.

The other four processes (PoS tagging, morphosyntactic analysis, lemmatization, and syntax analysis) remained unseparable, however. To integrate dapipe in the TT, we therefore created an aggregate content type and a 'splitter' tool that does nothing but outputting just one of the four annotation layers in the aggregated content type: **PoS tags**, **morphology** (later in the process to be combined into **msd**), **lemmas** or syntactic **dependencies**. Now we could segmentize and tokenize a TEI document, while keeping its TEI structure, feed it to dapipe, and retrieve the annotation layers as `<spanGrp>` elements in output TEI files. These annotation layers could then be merged into the input. This workflow design is illustrated in Fig. 3(a).

On inspection, the lemmas produced by dapipe were not very good. Not as good, we believed, as the lemmas that another integrated tool, CSTlemma, would have produced. We mapped the Universal PoS tagset, which dapipe employed, to the PoS tag set employed by CSTlemma. When that had been implemented in the TT, workflow designs that contained both dapipe and CSTlemma were shown in the list of proposed workflow designs. One of these is shown in Fig. 3(b). So now we had a workflow design using dapipe for all annotation layers and another workflow design that ignored dapipe's lemma output, using CSTlemma lemmas instead. Comparing the two outputs, it became clear that CSTlemma produced better lemmas than dapipe. We discovered another difference between the lemma predictions: whereas CSTlemma computed the lemma of a word according to rules dictated by the PoS tag assigned to that word, dapipe, when computing the lemma, seemed unaware that another part of dapipe was predicting the PoS tag of that word. The result was that a word could be PoS tagged as a verb, but lemmatized as though it was a noun.

It is in such experimental phases of projects that the TT shows its advantage over hand-made workflow designs: without a considerable investment of human working hours, we were able to compare two workflow designs before deciding which one to use to annotate all documents in the Danish corpus. On the one hand we had results produced by dapipe only, with lemmas and PoS tags predicted independently, and on the other hand we had results produced by dapipe and CSTlemma in unison, with lemmas created conditioned by the PoS tags produced by dapipe. On beforehand, it was impossible to know which result would be the best one. Dapipe's lemmas could be erroneous, but would be erroneous independently of any errors in dapipe's PoS tags. CSTlemma's lemmas could be more or less erroneous than dapipe's. If there were lots of PoS tagging errors, CSTlemma's results would very likely be worse than dapipe's, but if there were few PoS tagging errors, the lemmas would be of higher quality. No programmatic method would be able to predict which approach would be the best one. Only experimentation could tell.

We did a third experiment. We reasoned that if dapipe makes many PoS tagging errors, and if there is an alternative PoS tagger that makes fewer errors, than we should feed CSTlemma with PoS tags created by the alternative PoS tagger. Skimming through the list of workflow designs that the TT proposed, we found a workflow design that incorporated the Brill tagger, see Fig. 3(c). We run the same test document through all three workflows and compared the lemmas. As expected, workflow 3(a) and workflow 3(c) produced lemmas that not always were congruent with the PoS-tags predicted by dapipe. Furthermore, workflow 3(b) showed few lemmatization errors, compared to the other two, and many of the errors were due to PoS-errors. Since PoS errors percolate to the the syntax analysis, it seemed reasonable to also let the PoS errors percolate to the lemmatization, especially if the original dapipe lemmas had more errors than CSTlemma lemmas. After comparing the three workflow designs, we selected the workflow design depicted in Fig. 3(b).

The Danish Parliamentary corpus contains 688 xml files and cover a period from October 2014 until September 2020. We will annotate these files in groups comprising one year of debates at

a time, which amounts to about 100 documents per group. We expect that each group takes about 2 hours to be completely processed.

During the test phase of the workflow, we were in need of a visualisation of the results. The correctness of the syntactic annotation in the TEI P5 formatted workflow output was very hard to check, so we added an extra tool to the TT that transforms the hard-to-read TEI P5 format to a plain, easier to read CONLL-U format. To make the validation even more easy, we added another tool to the TT, a tool that converts the CONLL-U format to the Penn Treebank bracketed list format, which displays the syntactic dependencies as a tree structure.

Given that the TT was to have a workflow that enriches ParlaMint TEI P5 textual input with PoS, morpho-syntactic, lemma and syntax annotations, it was interesting to see that the same workflow designs could be applied to non-ParlaMint texts. From an earlier project, we knew that there are users who use the TT to annotate TEI P5 documents with PoS tags and lemmas, while retaining many sorts of TEI P5 tags attached to words in the input file, such as `<add>`, `<app>`, `<del>`, `<ex>`, `<lem>`, and `<rdg>`. Now these users in addition can enrich their texts with syntax annotations.

6 An Improvement that is not coming

There is an improvement that is hard, if not impossible, to deliver. The Text Tonsorium has the technical expertise to construct workflow designs that ... work, but it does not have the expertise to tell which workflow design is the best in a given situation.

In other workflow management systems, users rely on the expert knowledge of an experienced colleague who manually picks the tools that, together, constitute the best workflow design for a given task. End-users do therefore not have to choose between lemmatizer A and lemmatizer B. Not only that makes life easy for users, a hand-made workflow design can be used again and again, given a name, described and shared with other users. There are however some pitfalls:

1. Some tools may have changed, delivering output that is not quite the same as when the workflow was manually designed. The quality label may loose its credibility over time.

2. Some tools may improve so much that the same expert user would choose that tool instead of the one that was chosen as part of the manually edited workflow design. So end-users do not reap the fruits of technological progress.

3. Tools may reach end-of-life and stop being executable. Workflow designs that depend on such tools stop functioning as well. Reproducability of earlier experiments suddenly stops, and an expert user (the same or a new one) has to be called in to design a new workflow.

4. Workflow designers may for whatever reason choose to design several workflows that attain the same goal. The end-user would then have to choose the workflow that fits the actual situation best. That, of course, weakens the utility of having expert human users whose task it is to make choices on behalf of end-users.

5. Expert users may shun away from parts of the solution space for various reasons, but those reasons need not always be valid in situations that end-users encounter:

   (a) An expert user may disregard some tools for reasons that have little to do with their function, just to be on the safe side. For example, a tool may be known to be excessively resource gobbling, or slow, or unable to handle large input, or in an unstable beta phase of development. However, such problems may be of no importance for an end-user, hardware resources may become better and software may become available in a more stable version.

   (b) An expert workflow designer has to make assumptions about the input data that perhaps do not apply in the end-user's setting. For example, the expert may assess the quality of tools by running them with a text corpus that is an homogeneous mix of

many text types as input, while the end-user may work with text types with peculiar characteristics, such as bad spelling, OCR-errors, social media jargon, sociolects, or transcribed speech.

(c) The expert just does not think that very complex workflow designs are worth testing. The idea that simple, short workflow designs are to be preferred can steer a human expert towards suboptimal solutions.

The TT displays all viable workflow designs that fulfill the user's goal. There is little that can be done about this. The accumulation of metadata about the tools that together constitute a workflow design does not sum up to an overall quality assessment of the workflow design. The same tool can be considered a strong link in one workflow design, but a weak link in another one, so there is not a 'tool quality measure' that can be used in the computation of an 'workflow quality measure'. The only way to improve on this is to add metadata that encompasses two or more tools at the same time.

7 Coming Improvements

The TT has the potential to generate thousands of useful workflow designs. In order to make this an advantage over WMSes that require manual composition of workflow designs, it is necessary to guide users when they state their goal, so that they are not overwhelmed by too many designs to choose from. There are only a few drop down lists that the user has to consider, but specifying a goal can still be hard, since some of the values to choose from are relatively arcane.

The leading idea of the improvements we envision is that the specification of the goal can be done in easy steps. For each choice to be made, users will be guided by explanations and examples. Users who are already acquainted to the TT can skip this guidance, if they like.

Firstly, we want to find equivalents of the concepts that are used in the TT in the CLARIN Concept Registry (CCR). The TT can use the CCR descriptions for those concepts for which approved CCR equivalents exist. If we find that a candidate CCR concept would have been the perfect choice, we will communicate that finding to the Concept Registry Coordinators. If the TT needs a concept for which no CCR equivalent exist, we will try to make that a CCR candidate ('medieval', 'OCR', 'blackletter', 'pruned').

Secondly, we want to construct an assortment of output samples that we can show to users, so they can make better informed choices.

The TT is not able to compute workflow designs for all goal specifications that the user can make. Whether or not any workflow designs exist for a given input and goal can only be found out by letting the TT try to find those candidates. As a third improvement to the user experience, we need to inform users that an empty list of workflow designs is not a shortcoming and does not mean that there is a bug in the TT. An informed user whose goal cannot be fulfilled by the TT, might even appreciate that he or she did not have to spend hours on trying to manually compose a workflow design that cannot be composed, given the current set of available tools.

A fourth improvement will be the possibility to bookmark a workflow design. To bookmark a selected workflow design, the user will have to provide both a name and a reason for marking the workflow design, if possible comparing it with other already bookmarked workflow designs. Shared bookmarks make it easier to cooperate in a project and are useful as a reminder for future use. Bookmarks cannot be guaranteed to exist forever, however. Changes in the registered metadata of a tool can potentially invalidate a bookmarked workflow design. If that is the case, the bookmark must be marked obsolete and deleted after a grace period of for example two years. When a user has uploaded input to the TT, only bookmarked workflow designs will be shown that can be applied to that input.

An alternative to a bookmark is simply freezing and saving the workflow design itself. Technically, that would not be very different from bookmarking a workflow design, but conceptually there is a difference: a saved workflow design exists until someone deletes it. It can continue to exist, even if the TT has evolved so much that the saved workflow design no longer can be

executed. A bookmark is merely a pointer with some metadata attached to it. From their experience with bookmarks to websites, users do not expect that bookmarks to workflow designs can be used in all eternity. Since the TT is meant to be expanding and improving over time, bookmarks are more useful than frozen and saved workflow designs.

Researchers and students from the Faculty of Humanities at the University of Copenhagen will be involved in testing and improving the new interface to the TT.

8 Availability

There is an online version of the TT at the address `https://cst.dk/texton/` that can also be reached via the clarin.dk website (`https://clarin.dk/clarindk/tools-texton.jsp`). An identical version can be downloaded from GitHub[6]. Some of the tools in the GitHub repository are merely wrappers for open source tools that must be obtained separately, such as LibreOffice (which is used for conversion of office formats to RTF) and the OCR programs Cuneiform and Tesseract.

The TT can be installed on a personal computer, for example in the Windows Subsystem for Linux. Such an instance of the TT even runs while the computer is cut off from the internet and can then be used for handling very sensitive input, e.g. non-anonymized juridical documents.

9 Conclusion

We have in a few words described the current state of the Text Tonsorium, a Workflow Manager for NLP tools that has been re-launched to the CLARIN community. The Text Tonsorium automatically computes workflow designs that fulfil the user's goal and excutes the ones that the user selects. To illustrate that the final, sometimes hard, decision which workflow design to use, has to be taken by the user and cannot be automated away, we have devoted a section to how a recent project's aims were backed by the Text Tonsorium, suggesting several workflow designs, each with their advantages and disadvantages. Finally, we discuss how we hope to improve the user experience of the Text Tonsorium. We would be pleased to receive more suggestions from users and testers in the CLARIN community.

References

Tomaž Erjavec, Ştefan Bruda, Ivan Derzhanski, Ludmila Dimitrova, Radovan Garabík, Peter Holozan, Nancy Ide, Heiki-Jaan Kaalep, Natalia Kotsyba, Csaba Oravecz, Vladimír Petkevič, Greg Priest-Dorman, Igor Shevchenko, Kiril Simov, Lydia Sinapova, Han Steenwijk, Laszlo Tihanyi, Dan Tufiş, and Jean Véronis. 2010a. MULTEXT-east free lexicons 4.0. Slovenian language resource repository CLARIN.SI.

Tomaž Erjavec, Ivan Derzhanski, Dagmar Divjak, Anna Feldman, Mikhail Kopotev, Natalia Kotsyba, Cvetana Krstev, Aleksandar Petrovski, Behrang QasemiZadeh, Adam Radziszewski, Serge Sharoff, Paul Sokolovsky, Duško Vitas, and Katerina Zdravkova. 2010b. MULTEXT-east non-commercial lexicons 4.0. Slovenian language resource repository CLARIN.SI.

Erhard W. Hinrichs, Marie Hinrichs, and Thomas Zastrow. 2010. Weblicht: Web-based LRT services for German. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden, System Demonstrations*, pages 25–29. The Association for Computer Linguistics.

Bart Jongejan and Hercules Dalianis. 2009. Automatic training of lemmatization rules that handle morphological changes in pre-, in- and suffixes alike. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, volume 1, pages 145–153. Association for Computational Linguistics.

Bart Jongejan. 2013. Workflow management in CLARIN-DK. In *Proceedings of the workshop on Nordic language research infrastructure at NODALIDA 2013; May 22-24; 2013; Oslo; Norway. NEALT*

[6] `https://github.com/kuhumcst/texton`, `https://github.com/kuhumcst/texton-bin`, `https://github.com/kuhumcst/texton-linguistic-resources` and `https://github.com/kuhumcst/DK-ClarinTools`

Proceedings Series 20, number 89, pages 11–20. Linköping University Electronic Press; Linköpings universitet.

Bart Jongejan. 2016. Implementation of a workflow management system for non-expert users. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 101–108, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Lene Offersgaard, Bart Jongejan, and Bente Maegaard. 2011. How Danish users tried to answer the unaskable during implementation of clarin.dk. In *Proceedings of SDH*, November. SDH 2011- Supporting Digital Humanities.

Milan Straka and Jana 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.