Pregledni članak

Architecture for Editing Complex Digital Documents

Tomaž Erjavec
Department of Knowledge Technologies
Jožef Stefan Institute
Jamova cesta 39, Ljubljana, Slovenia
tomaz.erjavec@ijs.si

Summary

In several on-going projects we were faced with the dilemma of how to reconcile our goal of delivering standardly encoded digital editions, yet have the actual editing and annotation performed by researchers and students who had no knowledge of XML and the Text Encoding Initiative Guidelines (TEI), and, for the most part, no great interest in learning them. The developed solution consists of allowing the annotators use familiar and flexible editors, such as Microsoft Word (for structural annotation of documents) and Excel (for word-level linguistic annotation) and automatically converting these into XML/TEI. Given the unconstrained nature of such editors this sounds like a recipe for disaster. But the solution crucially depends on a dedicated Web service, to which the annotators can up-load their files; the service then, via XSLT scripts, converts them to the canonical encoding in XML/TEI, and from it back to a visual format, either HTML or Excel XML. These files are returned to the annotators, giving them immediate feedback about the quality of their encoding in the source, and can thus correct errors before they accumulate. The paper describes the web service and details its use in a project compiling a digital library and lemmatised corpus of XIXth century Slovene books.

Key words: production of digital editions, collaborative work, standards for text encoding, XML, Text Encoding Initiative, Web services

Introduction

A valuable part of digital heritage constitute historically important texts, so called text critical editions, typically comprising the facsimile, and one or more transcriptions, along with commentary. Furthermore, such text can be linguistically annotated, say lemmatised or part-of-speech tagged, for easier access and retrieval. They are thus multimedia editions, with a rich and complex structure, comprising metadata, editorial interventions, extensive cross-linking, and fine-grained annotations.

It is today generally assumed that such scholarly annotated digital texts must be stored in XML to ensure longevity as well as platform and software independence, ensuring the preservation of the extensive intellectual effort that went into their production. It is also accepted by many of the producers of such editions that the XML vocabulary for annotating the editions should be based on the TEI Guidelines (Sperberg-McQueen and Burnard, 2002), which have become the de-facto standard for constructing the XML schemas for a broad range of scholarly digital texts.

But while XML is well-suited for machine processing, it is less than ideal for authorial or editorial interventions into the text, esp. when used with the complex TEI-derived schemas. It is of course possible to edit XML documents directly in a plain text editor or, better yet, in specialised XML editors that support on-the-fly validation against a schema and schema-dependent drop-down menus. But, depending on the text type and required manual interventions, these generic editors are too clumsy for extensive work, and do not enable complex constraints on the allowed content and changes in the annotations. An additional problem with using XML editors for editorial work is the fact that many humanities scholars or students who are most likely to be doing this work have no knowledge of XML or TEI or any experience in editing it. This problem becomes all the more relevant in collaborative projects in which, say, a large number of students are hired to annotate a certain text or text collection. The effort required to first teach them how to use an XML editor and the underlying concepts might be prohibitive, and saving time necessary to perform each editorial intervention is essential.

Such problems of manual intervention can be resolved as they had been before the advent of XML: by developing specialised editing programs for the task at hand which store the data in the required format, and are optimised to perform validity checking and enable fast and easy editing of particular texts types or annotations. But such development is expensive in programming time, (human) editors need to be taught the specifics of the system and last but not least, the program might need to be installed on many different computers or on a Web server for which an uninterrupted internet connection is required.

On the other hand, standard desktop office editors are very versatile, can be easily configured for particular tasks, and most computer users are literate in their usage and already have them installed. In the context of scholarly text editing and annotation, two editors are especially relevant:

- Microsoft Word, the most used text editor, allows easy text authoring (e.g. spell-checking) and editing (e.g. hot keys), definition of complex document structures (sections, tables, notes), has good support for Unicode, allows the inclusion of graphics, etc.; and
- Microsoft Excel, a widely used spreadsheet editor, allows sorting, content-dependent formatting, cell protection, drop-down menus, multiple sheets, etc.

Both editors have applications in scholarly digital texts production: Word is appropriate for text authoring and editing and simple alignments (e.g. between the transcription and its facsimile), while Excel can be usefully employed for linguistic annotation, especially lemmatisation and part-of-speech tagging.
The missing link is, of course, the transformation from the file format of the respective editor to the format defined by a specific (TEI) XML schema. The implementation of such a transformation brings about two problems:

- how to parse complex input files the formats of which are under the control of a software company, hence without the necessarily accessible specifications and with no guarantees on modifications from one version of the software to another; and
- how to reconcile the very free ("visual") structure allowed by the editors to the much stricter ("semantic") XML schema controlled output.

This paper explains how we overcame these two hurdles in the otherwise appealing scenario: humanities editors are free to use tools they are familiar with (Word, Excel), no time investment into project specific (computer) editors is required, yet the final digital edition is stored in a standardised, well-documented and processable format, the TEI XML.

The conversion architecture

The conversion architecture is centred on a Web service which takes as input (possibly a combination of) Word and Excel documents and returns XML and HTML documents. The conversion consists of:

1. parsing the input documents;
2. converting (and merging) them to a TEI document;
3. validating the TEI document;
4. converting the TEI document to HTML;
5. (converting the TEI document to Excel);

Parsing the input

The first hurdle mentioned above, parsing – and making sense of – the vendor-encoded input documents, would have been much more difficult (if not impossible) to overcome some years ago, as we would need to develop or obtain software to parse native editor formats (RTF, Excel) or fall back on the lowest common denominator (plain text, tab separated file). We would also have to deal with significant problems of character set encodings, especially problematic with historical documents. Now, however, many applications offer XML formats for their data. Both Word and Excel (at least in the Office Professional edition) support saving and opening such documents as XML. Encoding problems with well-formed XML do not exits, as its native character set encoding is Unicode. Having XML as the input format allows for formally validating each input file but also allows – even in the absence of public and well-defined formal specifications, i.e. XML schemas – a window into the source format, as

XML documents are easier to reverse-engineer than fully-proprietary encodings. Possible changes in the format between versions are hence also relatively easy to accommodate.

Up-conversion to TEI and validation
The conversion from the proprietary XML to the TEI-compliant XML typically consists of a pipeline of XSLT transformations. The source XML is first converted into a simple TEI and from there into the project-required TEI encoding. The Extensible Stylesheet Language Transformations (XSLT) is an XML-based language used for the transformation of XML documents into other XML or "human-readable" documents, such as HTML.
The formal, “syntactic” validation of the produced TEI document is straightforward, as the TEI document can be checked by any validating XML parser against the project specific TEI schema.

Down-conversion to HTML
The canonical TEI document is then converted, again with XSLT transforms, into a “readable” version of the document. On the on hand, this has to be done for end-users (of digital libraries), but it is also of crucial importance in overcoming the second hurdle mentioned above, i.e. how to reconcile the unconstrained and presentation-oriented nature of documents, especially the ones created in Word, with the strict and interpretative TEI schemas. Namely, the XSLT generated HTML format, esp. with its table of content, various indexes and other cross-references and use of HTML formatting features, such as colour, gives the humanities editors the feedback they need in order to validate whether their source documents are indeed well-formed; only if the structure and annotations are correct in the produced HTML, are they valid in the source. While, as will be discussed below, good guidelines are still needed, the proposed approach also enables self-correction and self-teaching of editors who ultimately produce an exact TEI document.

Down-conversion to Excel
In certain scenarios we also generate Excel (XML) documents which then serve as input to the editing process, and are uploaded to the server after they had been corrected. Such automatically generated Excel documents can be quite sophisticated using a simple trick: a template Excel document is created by hand, with certain cells containing “hooks”, and stored as XML. The conversion then takes this template and replaces the hooks with actual data from the TEI document, duplicating the rows as necessary.

The Web Service
The implemented web service runs under Linux/Apache, using CGI/Perl. The Perl script:

1. takes the uploaded file, possibly compressed, with the archive containing multiple files;
2. calls various transformations with user-selected parameters;
3. returns the result, either directly via HTTP or as an archive file; and
4. logs each transaction, possibly archiving the input and output files.

The presented architecture has thus the following characteristics:

- it enables the editors to work with familiar and powerful tools yet produces TEI conformant output;
- it allows for a gradual learning process and step-wise refinement of the target documents;
- the data is standards-compliant (TEI, XML, (X)HTML, XSLT);
- the software components are Open Source, (Linux. Apache, Perl, libxml); and
- it is not very difficult to modify for new projects.

The AHLib project

So far, the presented web service has been used in several projects / editions. In the initial attempts the Web interface supported only uploading of a Word file and then displayed the derived TEI and HTML files. In this setting, Word is used primarily as an authoring environment. Experience has shown that quite exact guidelines are needed to enable the production of sufficiently constrained Word documents to allow for further processing. This is why editors in later projects have been given short courses, as well as written, quite specific guidelines about what and how to annotate the source document, together with a Word dot file containing the styles used in the project.

In this section we present the Web interface used to compile the AHLib digital library / corpus of XIXth century Slovene books. This is still work in progress, although the Web interface has already been implemented, as well as a "debug" version of the TEI to HTML conversion.

In AHLib each book is represented by the facsimile and a structured diplomatic transcription, hand-corrected from OCR. The text is automatically lemmatised, using the methods described in Erjavec et al. (2005), and then corrected manually. The AHLib Web interface is thus used for correcting two types of errors. The first are errors in the text itself for which each book must be proof-read from the OCR original. At this stage text structure is annotated as well, e.g. headings (divisions), footnotes, figures, page breaks (for alignment with the facsimile), foreign language passages, critical corrections (in case of typos in the original), etc. The second type of errors concerns linguistic annotation. Each word token in the text is first automatically lemmatised and then this lemmatisation is corrected by hand.

The set-up and the intended workflow in this application are rather complicated, mainly due to the fact that there is no simple way of splitting these two annotation types into two separate stages. In particular, checking the lemmatisation often reveals overlooked OCR errors in the text which can only be corrected by going back to the RTF. A further problem is that the automatic lemmatiser (a machine learning program, coupled with a trainable tagger) has been trained on contemporary Slovene which differs considerably from the (non-standardised) language of a century ago. Therefore the lemmatiser consistently makes errors with certain archaic words.

We solved these problems by splitting the process into three stages, allowing for multiple file input and output, and up- or downloading partially corrected files:

1. The user uploads the RTF file and receives either the TEI or HTML; an example of the produced HTML is given in Figure 1. This stage is appropriate for structuring the document and initial proof-reading.
2. The user uploads the RTF or TEI file which is automatically lemmatised and the (structured and) lemmatised version returned as TEI or HTML; an example lemmatised HTML file is given in Figure 1. The lemmas are also furthermore checked against a large contemporary dictionary of Slovene. The unknown lemmas are returned with word-forms and context from the corpus as an Excel table, as illustrated in Figure 2. This table is then manually checked: in case a word is a typo, it is corrected in the RTF file and deleted from the Excel table; if a word is lemmatised incorrectly, its lemmatisation is corrected, and the corrected Excel table is uploaded to the service where it serves as a gold-standard lexicon for the lemmatisation of further texts. It is also possible to perform this process cyclically by submitting the RTF/TEI files and the (partial) Excel table of unknown words together.
3. The user uploads the RTF or TEI documents, and the complete lemmatised text is returned as an Excel table, as illustrated in Figure 3. The user has to check / correct the lemmatisation of each token in this table, and finally submit the RTF/TEI together with the corrected Excel in order to arrive at the final structured and linguistically annotated TEI document. Again, cyclic improvement is possible by submitting the RTF/TEI together with partial Excel. This step is slightly more complicated as Excel imposes an upper limit of 64,000 rows per table, while a book can have more than that number of words. We therefore support the download of multiple Excel files, each containing a portion of the book. Furthermore, the user has the option of retrieving the Excel in the text order or sorted alphabetically.

Conclusions

The paper has presented an environment for manual interventions into XML-based scholarly editions. The basic assumption is that it easier for au-

thors/editors/annotators to use generic and readily available editors than to edit the XML, as well as faster for computer linguists to write or modify XSLT scripts than to develop specialised editors for particular projects. The architecture relies on a Web service that transforms input documents into a standardised format, validates them syntactically, and returns them for semantic validation. The approach was illustrated in a setting in which Word and Excel are used for authoring or correcting the base text and word-level linguistic annotation respectively.

Our experience with the presented approach shows that it is important to give annotators a tutorial and detailed guidelines, and that the approach is mostly appropriate for shallow encodings. For example, trying to unambiguously represent nested annotations in Word (e.g. a correction consisting of a deletion and addition) or cross-references is very difficult, and complex XSLT transformations are needed to transform this information into TEI. We therefore see the usefulness of this approach esp. in collaborative projects with each annotator investing minimal time in training and annotation. Such an annotation scenario is becoming increasingly popular in the HLT community, and wider. So, for example, Mihalcea and Chklovski (2004a, 2004b) describe their "Open Mind Word Expert" site where e.g. student contributors are presented with a set of natural language (e.g., English) sentences that include an instance of the ambiguous word and are asked to indicate the most appropriate meaning with respect to the definitions provided, thus building a word-sense disambiguated corpus. Similarly, Good et al. (2006) report on an experiment where volunteers untrained in knowledge engineering developed a partial ontology via a Web interface. In a non-HLT context, a distributed approach to annotating is used by the Mechanical Turk (http://www.mturk.com/) by Amazon (also dubbed "Artificial Artificial Intelligence") where humans are paid to classify instances of e.g. pictures or texts into predetermined categories. Such tasks are posted to the Turk by companies that need large annotated datasets, typically for training machine learning systems. Both examples above are much more sophisticated than ours in terms of the number of users they allow, but much simpler in the kinds of annotations they envisage – the main difference is that they directly employ a web interface, while our architecture assumes off-line editing in Word or Excel.

As could be noticed, our approach is built around the notion of open standards and software, so a legitimate question is why we have opted to support Microsoft Word and Excel, rather than their open source Open Office (http://www.openoffice.org/) equivalents, OO Writer and OO Calc. The reason is simple: most of the editors and annotators that we worked with have Microsoft Office already installed on their computers, and are reluctant to install the OO suite and learn to use it. Also, in our experience, OO tools still lag behind Microsoft in terms of usability. However, the use of the XML-based OpenDocument standard as the native format for OO applications is a significant ad-

vantage, so we might reconsider our decision in future version of the Web service.

In our further work we also plan to address the question of version control, which is currently lacking in our system, to enable multiple editors to correct a set of documents without the danger of conflicts.

References

Erjavec, T., Ogrin, M. (2005) Digitalisation of literary heritage using open standards. In Paul Cunningham, Miriam Cunningham (eds.). Innovation and knowledge economy: issues, applications, case studies, (Information and communication technologies and the knowledge economy). Amsterdam [etc.]: IOS Press, 999-1006.

Erjavec, T., Ignat, C., Pouliquen, B., Steinberger, R. (2005) Massive multilingual corpus compilation: ACQUIS Communautaire and totale. In Proc. of the Second Language Technology Conference. April 2004, Poznan.

Good, B. M., E. M. Tranfield, P. C. Tan, M. Shehata, G. K. Singhera, J. Gosselink, E. B. Okon, and M. D. Wilkinson. (2006) Fast, Cheap and Out of Control: A Zero Curation Model for Ontology Development Pacific Symposium on Biocomputing 11. http://psb.stanford.edu/psb-online/proceedings/psb06/good.pdf

Mihalcea, R. and T. Chklovski, Teaching Computers: Building Multilingual Linguistic Resources with Volunteer Contributions over the Web, in The LISA Newsletter - Globalization Insider, September 2004. http://www.lisa.org/archive_domain/newsletters/2004/3.3/mihalceaChklovski.html

Mihalcea, R. and T. Chklovski, Building Sense Tagged Corpora with Volunteer Contributions over the Web, book chapter in Current Issues in Linguistic Theory: Recent Advances in Natural Language Processing, Editors Nicolas Nicolov and Ruslan Mitkov, John Benjamins Publishers, 2004.

Sperberg-McQueen, C. M. and L. Burnard, (eds.) (2002). Guidelines for Electronic Text Encoding and Interchange, the XML Version of the TEI Guidelines. The TEI Consortium, http://www.tei-c.org/

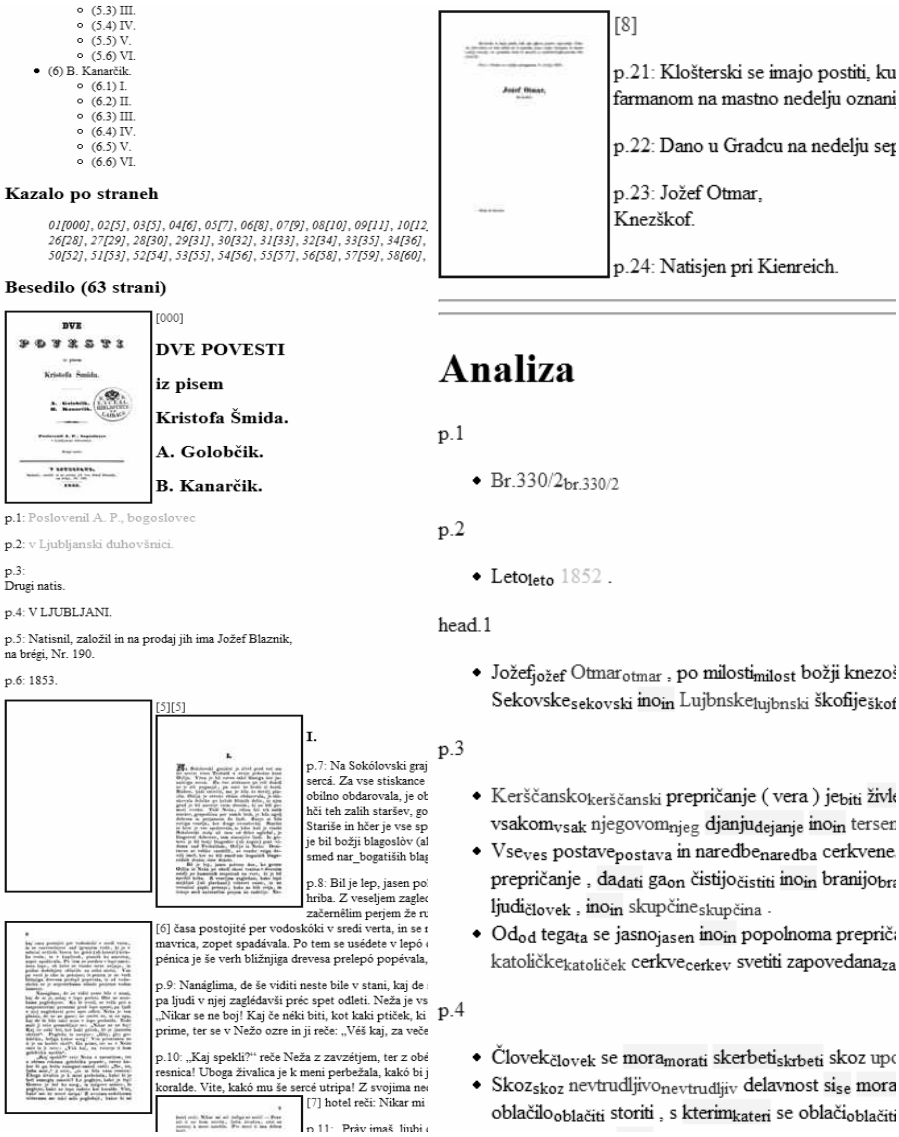

Figure 1. Two screenshots of the HTML view, first one from the beginning of the book, giving the end of table of contents, index by pages, and start of text containing facsimile and the transcription. The second one gives the end of the text, and the start of the linguistic analysis, i.e. the lemmatisation – only lemmas different from the word-form are shown. The colour indicates the status of the correction status of the lemma.

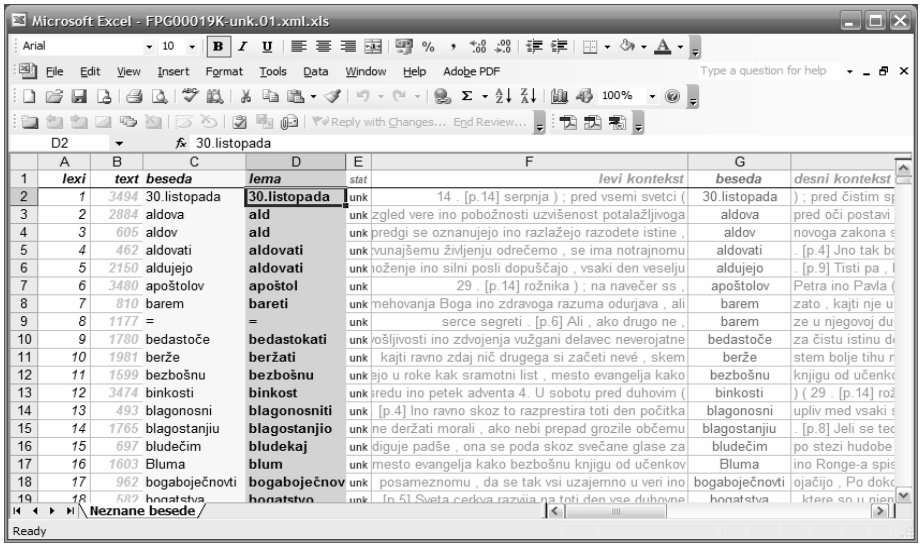

Figure 2. Excel spreadsheet for first round of corrections, giving only the words with out-of-vocabulary lemmas. Column A gives the lexical sort order of the table, B the text sort order, C the word-form, D the lemma (this is the column that annotators correct), E the status of the lemma (here only unknown), and F,G,H the concordance of the word.

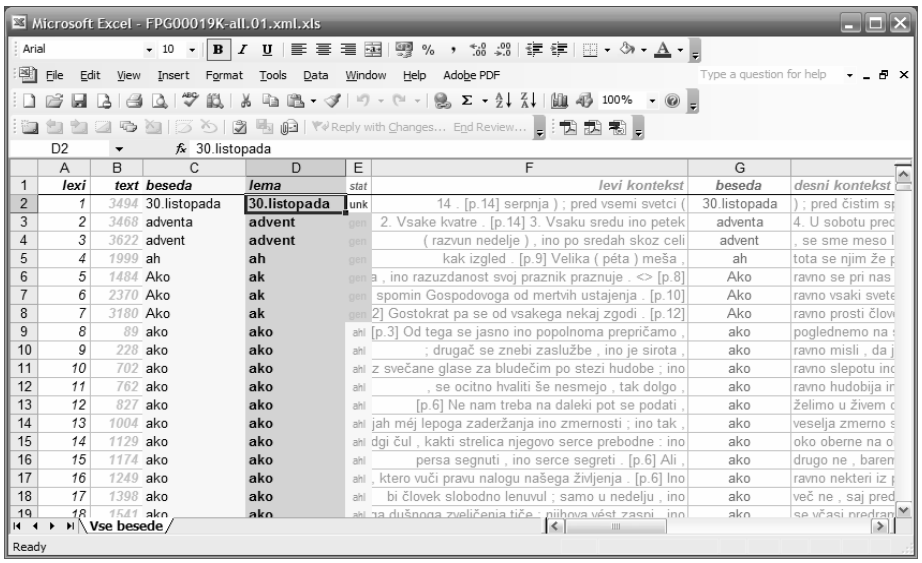

Figure 3. Excel spreadsheet for final corrections, giving the lemmatisation of all word tokens in the text. Here the status (E) of the word can be, in addition to unknown lemmas, also that of general lemmas, lemmas added to the AHLib specific lexicon, and ambiguous lemmas (not shown in the example).