Pacific Symposium on Biocomputing 12:391-402(2007)

SEMI-AUTOMATED XML MARKUP OF BIOSYSTEMATIC LEGACY LITERATURE WITH THE GOLDENGATE EDITOR*

GUIDO SAUTTER, KLEMENS BÖHM

Department of Computer Science, Universität Karlsruhe (TH), Am Fasanengarten 5 76128 Karlsruhe, Germany

DONAT AGOSTI

Division of Invertebrate Zoology, American Museum of Natural History, New York NY 10024-5192, and Naturmuseum der Burgergemeinde Bern, 3005 Bern Switzerland

Today, digitization of legacy literature is a big issue. This also applies to the domain of biosystematics, where this process has just started. Digitized biosystematics literature requires a very precise and fine grained markup in order to be useful for detailed search, data linkage and mining. However, manual markup on sentence level and below is cumbersome and time consuming. In this paper, we present and evaluate the GoldenGATE editor, which is designed for the special needs of marking up OCR output with XML. It is built in order to support the user in this process as far as possible: Its functionality ranges from easy, intuitive tagging through markup conversion to dynamic binding of configurable plug-ins provided by third parties. Our evaluation shows that marking up an OCR document using GoldenGATE is three to four times faster than with an off-the-shelf XML editor like XML-Spy. Using domain-specific NLP-based plug-ins, these numbers are even higher.

1. Introduction

Today, there are model organisms with huge bodies of literature increasingly digitally available. The descriptions of the remaining 1.5 Million known species alone, however, is scattered in an estimated 10 to 100 Million pages of printed record in thousands of journals and books. This biosystematics literature is in a very unique situation within the entire body of literature. Large parts of it are in a highly standardized structure, e.g. treatments or keys. They contain information in a very concise form, which is not available anywhere else. Its main sections comprise the descriptions or treatments of the species (including character X species data matrices, images and distribution records), tools for identification (keys), and phylogenies. A description of a species is comparable to its DNA sequence only at a higher organizational level (11). Therefore this body of literature offers a unique chance for data extraction and mining for

* Work supported by grant BIB47 of the Deutsche Forschungsgesellschaft and grant GRT963055 of the National Science Foundation.

biomedical and life sciences when it is transformed into a machine-readable form. Since all the data in a publication belongs to a particular species or higher level taxon, the insertion of markup identifying the taxonomic names and descriptions transforms a text document into a record of a biodiversity database. The taxon name serves as the unique identifier. Some of these databases are currently implemented, e.g., ispecies.org, INOTAXA, or Encyclopedia of Life (12). They target the integration of existing biodiversity data from different sources. This in turn allows mining and linking data from currently disconnected data sources such as genomics, behavior or distribution data. Figure 1a shows an example document as OCR produces it (idealized, no errors). Figure 1b shows the same document after the markup process. Obviously, only the markup enables a machine to read the information which species was collected in which location.

```
Ectatomma tuberculatum Olivier.- Sangre Grande, (R. Thaxter)
Ectatomma ruidum Roger.- Port of Spain, (R. Thaxter)
```

Figure 1a. A legacy document as OCR output (idealized, no character misrecognitions)

```
<treatment>
 <taxonName>Ectatomma tuberculatum Olivier</taxonName>.-
 <location>Sangre Grande</location>, (R. Thaxter)
</treatment>
<treatment>
 <taxonName>Ectatomma ruidum Roger</taxonName>.-
 <location>Port of Spain</loaction>, (R. Thaxter)
</treatment>
```

Figure 1b. The same legacy document after the markup process.

A first approach towards digital availability and integration is the community-wide initiative to scan and OCR the biosystematics literature deposited at the main US and UK natural history museums (Biodiversity Heritage Library: www.bhl.si.edu), growing taxon based (15) as well as commercial initiatives (16). This results in pdf and raw OCR-ed documents. The first step towards machine readability is a cleanup of OCR errors and artifacts originating from the print layout. The second step is the insertion of structural markup into the documents, often including a cleanup of structural XML markup inserted by the OCR. Further steps towards full machine readability of this body add semantic markup on different levels of detail, e.g., treatments and scientific names. Currently, the tools for all these steps are vanilla XML editors.

The only automation support for this process so far have been tools to find and extract scientific names (TaxonGrab (10), FAT (2), and FindIT(13)). These tools provide good results, but are hard to apply when using a common XML

editor. In this paper, we analyze the requirements on editors intended to support all the steps from OCR output to full machine readability: OCR cleanup, structural markup, NLP-based scientific name extraction, markup of the treatments. In particular, we focus on possible automations of the markup process. This is in order to reduce user effort as far as possible.

A major difficulty is the integration of manual text editing and NLP. This is because the former works on characters, while the latter usually works on sequences of tokens, i.e., regards words as the atomic units of a text. Finally, we present the GoldenGATE editor, which we have built to implement this difficult integration. Our evaluation shows that the markup process is three to four times faster if the user can make use of such a tight integration.

The rest of this paper is organized as follows: Section 2 presents existing editors and NLP tools, which can be useful for automated detail-level markup. Section 3 analyzes the markup process and specifies the requirements on an editing tool supporting this process. Section 4 presents the GoldenGATE editor, which is intended to comply with these requirements. Section 5 features experimental results, which demonstrate the feasibility of our new tool. In Section 6, we conclude with a discussion and an outlook to future work.

2. Related Work

In this section, we give an overview of existing tools and editors, which might be useful for the markup of legacy literature. We also point to some freely available NLP tools and libraries, which can be helpful for automating the markup process as far as possible. While manual markup is not desirable, fully automated markup solely relying on the NLP on the other hand is not feasible either: First, the markup accuracy required is higher than the 95 - 98 % provided by up-to-date NLP tools. Second, applying a sequence of such tools, which perform different parts of the markup process and build on the results of each other, is likely to result in a summation of the errors. Think of a noun-phrase chunker which builds on the output of a part-of-speech tagger. If the latter produces erroneous tags, the former is likely to produce erroneous output as well. A sequence of five such tools, for instance, is likely to have an accuracy of around $98\%^5 \approx 90\%$, which is less than required. Thus, there is a need for manual correction after each automated markup step (i.e., the application of one automated tool). This in turn requires an editor tightly integrating NLP-based automated markup functionality and manual editing and tagging.

2.1. *Editors*

Before we analyze the markup process, we discuss existing editors, which form its current basis, and are widely used for handling textual data in general.

General-purpose **text editors** like UltraEdit (5) are powerful editors for all kinds of text-based data, e.g., plain text, XML, or programming languages. Many of these editors natively provide syntax highlighting for XML and for common programming and script languages. Some also support recording macros for frequently used editing steps, and for including external components.

On the other hand, most text editors are general-purpose so that they do not provide any special support for XML markup of text, e.g., some sort of support for inserting XML tags. This is cumbersome and unnecessary, since the functional parts of the XML tags could be inserted automatically.

XML editors, like the widely used XMLSpy (3) and Oxygen (4), are built to support handling existing XML data: DTDs and XML schemas, the XPath and XQuery query languages, XSLT, etc. They also provide some automation in creating new tags. But with them, the natural sequence of actions seems to be building the structure first and then inserting the content. Inserting tags in existing content is not exactly supported well. Besides the query and transformation languages included, XML editors rarely provide mechanisms for automated changes to the content or markup. They are not designed to apply NLP because it is not a usual part of XML data handling.

2.2. *NLP Tools*

In the following, we describe some NLP tools which might be useful for automating the detail-level markup process of legacy literature.

The **OpenNLP** (6) project hosts a variety of smaller, mostly open-source projects related to the development of NLP tools. The tools provided are heterogeneous with regard to purpose, programming platform, and quality. Among others, the functionality provided comprises text tokenization, part-of-speech tagging, noun-phrase and verb-phrase chunking, named entity recognition, and semantic parsing. The former four build on each other and form the basis for the latter two. The latter are interesting for the detail-level automated markup, e.g., recognition and tagging of collecting sites, i.e., the location where a particular specimen has been collected.

GATE (7) is an NLP suite developed by the University of Sheffield. It offers functions comparable to OpenNLP, but also provides more complex processing, e.g., for co-reference resolution, and can produce XML output. The GATE suite also includes Apache Lucene (14) for information-retrieval purposes and a GUI for visualization. It is relatively easy to develop new

components and include them in the processing pipeline. On the other hand, the purpose of GATE is NLP research and automated evaluation rather than document markup and management. While an AnnotationDiff tool is provided for computing f-Scores from the results achieved with test corpora or evaluation tasks, GATE lacks any facility for editing the text and the markup of the documents manually.

LingPipe (8) is a professional NLP suite developed by alias-i. Apart from tokenization, which works with rules, almost all the analysis functions are based on statistical models (Hidden Markov Models **(9)**, in particular). This implies the need for training with pre-annotated data. Once a model is generated from the training data, it can be applied to annotate documents. The basic idea of LingPipe is that it can generate and apply models for a wide range of purposes. Ready-to-use models are available for part-of-speech tagging and named entity recognition. While LingPipe provides powerful NLP functionality, there is no user interface. Thus, it has to be included in another program to be accessible in ways different from the command line.

3. Requirements Analysis

In this section, we analyze the conversion process of digitized legacy literature: From OCR output to valid XML documents with detailed markup both regarding structure and 'semantically' important parts. Given this, we summarize the requirements on an editor that supports the user during this process. We then discuss some additional aspects influencing the design of an editor for markup of legacy literature.

3.1. *The Markup Process*

Figure 2 visualizes the digitization and markup process form printed legacy documents to XML data, as perceived by individuals involved in the process: First, a user will scan and OCR-process the printed documents. Even the best and best-trained OCR software achieves 100% accuracy, and old or low-quality source documents result in lower quality. This means that the output text will contain a significant number of misspellings and other character-recognition errors. With regard to later application of NLP tools, these errors are a serious problem. Named entity recognizers, for instance, often make heavy use of gazetteers, which will not be useful in the presence of misspellings. Misplaced punctuation marks are likely to disturb a tokenizer or sentence splitter.

Further problems arise from the page layout of the printed original, which can include footnotes, captions, page numbers, page border decorations, and so

on. The OCR is likely to mix these parts up with the main text so that the resulting text is inconsistent.

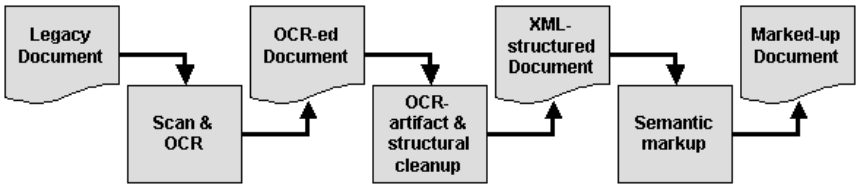

Figure 2. The process of marking up a legacy document

Consequently, if we want to apply NLP tools for automating parts of the markup, the process starts with the correction of OCR errors and layout artifacts. This also induces structural cleanups like re-concatenating hyphenated words, correcting paragraph borders, or moving captions to the end of the paragraph enclosing them. An editor should support a user in these actions as far as possible. Re-concatenating hyphenated words, for instance, is a cumbersome task if it is done one by one manually, but it automates easily. Punctuation and capitalization in turn allow checking the structure of paragraphs automatically, leaving only the ambiguous cases for manual intervention. Once the structural integrity of a document is restored, the next steps include the markup of semantic units, e.g., individual treatments, and meaningful parts, e.g., taxonomic names and collecting locations.

Since the latter are often locations, their markup can be automated: NLP provides powerful recognition algorithms like the one presented in (1), which can be applied for the markup of collecting sites. (2) presents an approach for high-accuracy taxonomic name extraction. Its output can serve as the basis for automated detail-level markup. Consequently, an editor intended for semantic markup of legacy documents should allow for integration of existing NLP tools. It should also provide lightweight interfaces for including additional tools so that the editor is easy to extend according to the particular automation needs of the user. Further, for similar documents, a user is likely to apply the same choice of automated tools in the same order, thus defining a sequence. For easier use of such a sequence, it is desirable to access it as one tool.

Despite all possible automations, manual editing is indispensable because NLP rarely achieves 100% accuracy. This is especially important where one NLP component builds on the output of previous ones: Erroneous input is likely to induce faulty conclusions, and the errors typically add up. Consequently, an environment supporting automated NLP-based markup also has to provide facilities for manual editing of both the text and the markup.

3.2. *Requirements*

Summarizing the transformation process, an editor intended for the XML markup of digitized legacy literature has to comply with the following requirements in order to assist its users as well as possible:

- Automation support for structural cleanup of documents,
- Easy manual editing of both text and markup,
- NLP support for automated markup,
- A lightweight interface for developing and including new NLP tools, according to the special needs of a specific application,
- Integrated access to sequences of tool.

3.3. *Additional Aspects*

In addition to the key features listed above, there are some other aspects worth consideration. Different OCR tools provide different types of additional information in their output. While some simply produce plain text, others insert generic XML tags, and yet others provide HTML formatted documents. Consequently, an editor should be able to make use of any formatting contained in a document, to unify it for subsequent steps, and still provide good automation if there is no formatting at all initially. During the editing process, it is desirable to use some unified, generic markup, which does not depend on an application-specific XML schema. This is because, first, it is not feasible to develop generic NLP tools dependent on a specific XML name space. Second, it may be desirable to transform the completed documents into a variety of application-specific XML schemas. Since different schemas provide different types and levels of detail markup, a direct inter-schema transformation may not be possible in the general case. If the input schema provides no markup for locations, for instance, a schema-transformation tool cannot introduce such markup. Thus, an editor needs to support different XML schemas, rather than only a specific one.

4. The GoldenGATE Editor

In this section, we present and describe the GoldenGATE editor, which we have built to comply with the requirements identified in the previous section. It combines automation-assisted markup and text editing with external NLP tools.

4.1. *The Document Editor*

In the GoldenGATE main window, each document has its own editor tab. We refer to such a tab as a document editor. The markup of legacy literature

includes editing both the XML markup and the document text. Experience with standard XML editors like XMLSpy shows that editing documents is cumbersome if there are too many XML tags. This is because the tags are in the way of a concise view on the textual content. On the other hand, editing XML markup is unnecessarily cumbersome in an editor that supports plain text editing as well. This is because the XML tags have to be inserted in the text character by character. An editor in turn could automatically produce the functional parts of the tags, i.e., the XML-specific characters around the element names and the attribute values. Consequently, the editing functions for XML markup on the one hand and for textual content on the other hand are distributed to two different editor views in GoldenGATE. Thus, a document editor has three sub-tabs: An annotation editor, a plaintext editor, and, in addition, an XML view, which provides no editing functionality, but a view on the document as nicely indented and laid-out XML.

The *annotation editor* (Figure 3) provides automation assistance for manual XML tagging. The buttons on the left invoke recently used functions, and user-defined ones, i.e., Macros. The checkboxes on the right allow showing and hiding individual tags by name. The annotation editor uses a token-based data model, which treats a word as an atomic unit, for two reasons: First, a user will insert XML tags between words in the very most cases. Second, almost all NLP tools work on tokens rather than on characters. Consequently, an interaction that is based on selected text, e.g., enclosing a passage of text in a new tag, will automatically affect complete words, even if only a part of them is selected.

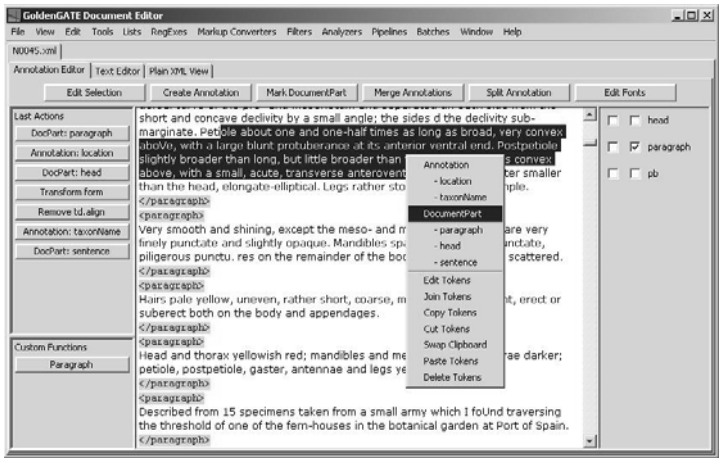

Figure 3. The annotation editor.

Enclosing a sequence of words in an XML tag is easy and intuitive in the annotation editor: First, select the words to enclose in the new tag. Second, right

click and select Annotation in the context menu. This will open a prompt for entering the tag name. In addition, the context menu offers the most recently used tag names for instant selection. The annotation editor also provides functions for joining and splitting tags. This is helpful for, e.g., correcting structural markup generated by OCR software. In addition, it contains functionality for automatically tagging paragraphs, and for automated cleanup of their inner structure, including the re-concatenation of hyphenated words.

The text editor provides editing functionality known from standard text processing tools. Because editing textual content is cumbersome if it is spread out between several XML tags, the text editor hides all tags to provide a concise view on the document content.

4.2. *Integration of External NLP Tools*

As discussed in Section 2, NLP provides powerful tools for extracting meaningful phrases and word sequences from text, which are well suited for detail-level markup of legacy documents. The GoldenGATE editor provides a lightweight interface for integrating external NLP tools.

To integrate an individual NLP component into the GoldenGATE editor, this component needs to implement the so-called Analyzer interface, or it needs an encapsulating wrapper, which implements this interface. The responsibility of the wrapper is to translate the token-based data model of the annotation editor to the data model used by the NLP component. Since most NLP tools work on token arrays representing the tokens as Strings, this tends to be straightforward. The wrapper also has to translate the output of the NLP component back into the data model of the annotation editor. Since most NLP components arrays of Strings to mark the extracted parts, this tends to be straightforward as well.

A complete suite of NLP tools can bind to GoldenGATE via a wrapper factory, which implements another interface. The task of such a factory is to wrap the NLP tools, so that the individual tools need not be wrapped manually. The most feasible way of using this binding method is to provide a factory wrapping a common super class of a set of NLP tools. Once a tool is wrapped to implement the interface, it can simply be packed into a jar file along with the wrapper. The jar file is subsequently stored in defined location where GoldenGATE will automatically detect and include the new tool once it is restarted. In addition, a user can trigger the search for new tools manually.

4.3. *Sequencing of Tools – Pipelines*

As mentioned in Section 3, a user is likely to apply the same NLP tools to similar documents in the same order. Thus it is desirable to access such a

sequence as one tool. GATE **(7)** uses a mechanism called Pipelines for this purpose. It bundles a sequence of tools and applies them in a specific order. Borrowing this idea, we have integrated Pipelines in the GoldenGATE editor. A GoldenGATE Pipeline is a sequence of external or built-in tools. When it executes, it invokes these tools subsequently. In contrast to GATE, users may configure a GoldenGATE Pipeline to display the documents for manual correction after being processed by each tool and before the next one is applied. This prevents the propagation of errors from one tool to subsequent ones.

4.4. *Additional Functionality*

In this section, we describe some additional features of the GoldenGATE editor.

Macros allow adding custom functions to the annotation editor. A macro combines first marking up the selection with a predefined XML tag, and then applying some automated processing to the content of the newly created tag. Two possible applications of this mechanism are (a) marking up a paragraph and applying structural normalization afterwards, or (b) marking up a treatment and then applying a tool that creates the internal markup of the treatment automatically, e.g., the 'nomenclature' and 'materials examined' domains.

Lists provide a straightforward way of applying gazetteers. GoldenGATE natively provides some lists (stop words, for instance), and new ones are easily added: GoldenGATE loads them from files or URLs, or extracts them from the documents, making use of existing markup.

Besides Macros and Lists, GoldenGATE provides a variety of further features, which we cannot describe here due to space limitations. These features include tagging functions based on regular expressions, or processing an entire folder of documents automatically.

5. Preliminary Evaluation

This section features a preliminary evaluation of the GoldenGATE editor, to quantify its benefit in marking up legacy literature.

5.1. *Experimental Setup*

Our evaluation criterion is the time it takes a user to mark up a document, starting with the unmodified output of an OCR tool. Our evaluation is as follows: A domain expert carries out the task, once using GoldenGATE, and another time using the 2005 version of the XML editor XMLSpy, to obtain a reference point. We stress that our test person is not a computer scientist. He has practical experience with vanilla XML editors, notably XMLSpy. As test documents, we use three issues of the American Museum Novitates (No 349

(1929), 1396 (1949), and 2257 (1966)), a life science periodical issued by the American Museum of Natural History. The original hardcopies have been scanned and OCR processed with Abbyy FineReader. The output contains HTML markup regarding fonts and structure, but a significant part of the latter is erroneous. The documents comprise a total of 24 pages (8 pages each) and 12 treatments (4 treatments each). The markup created during the test includes structural and semantic markup. The latter comprises treatments, taxonomic names, and collection events.

5.2. *Results*

Table 1 displays the time needed for the markup process with the two tools under evaluation. It turns out that GoldenGATE benefits significantly from the token-based, semi-automated data handling of the annotation editor. The NLP tools applied to the fine-grained markup of important parts (e.g., collecting sites) result in an even larger difference.

Table 1. Markup time from OCR output to fully marked-up document (in minutes, time spent with the individual test documents in brackets)

Editor / Markup Step	XMLSpy	GoldenGATE
OCR errors	9 (3, 4, 2)	4 (2, 1, 1)
Structure Cleanup	4 (2, 1, 1)	19 (7, 5, 7)
Treatments	84 (29, 31, 24)	7 (2, 3, 2)
Tax. names	28 (10, 10, 8)	5 (1, 2, 2)
Coll. events	24 (8, 9, 7)	9 (4, 2, 3)
Total	149 (52, 55, 42)	44 (16, 13, 15)

The only step that takes longer in GoldenGATE is the structural cleanup. This is due to different approach with the two editors: XMLSpy requires to first remove all the HTML and to insert new structural markup subsequently, while GoldenGATE transforms and corrects the existing tags. Thus the structural cleanup becomes an integral part of the structural markup. Summing up the structural cleanup and markup steps, GoldenGATE (19 + 7 = 26 minutes) is more than three times faster than XMLSpy (4 + 84 = 88 minutes).

6. Discussion

In this paper, we have introduced the GoldenGATE editor, the first editor specifically designed and built for the digitization of legacy biosystematics literature. It supports all the steps from OCR output to full machine readability: OCR cleanup, semi-automated markup (both structural and semantic), including the detection of treatment boundaries and the markup of the internal structure of treatments. It allows the application of automated external markup tools, like TaxonGrab (10), FAT (2), or FindIT (13) for the markup of scientific names.

Our evaluation has shown that the GoldenGATE editor simplifies and accelerates the markup process significantly. This advantage results from both the semi-automated, token-wise XML editing and the integration of existing NLP tools for automated detail-level markup.

We plan to include more functionality in the future, like better support for OCR error correction. We also intend to develop additional NLP tools for more markup automation on the detail level. This includes, for instance, automated detection and markup of morphological characters and states.

The current version of GoldenGATE is available at http://idaho.ipd.uni-karlsruhe.de/GoldenGATE/.

Acknowledgements

We thank our US collaborators Terry Catapano, Bryan Heidorn, Bob Morris, Neil Sarkar, and Christie Stephenson for their interest in our work and their valuable suggestions.

References

1. A. Mikheev, M. Moens, C. Grover, *Named Entity Recognition without Gazetteers*, in Proceedings of EACL, Bergen, 1999
2. G. Sautter, K. Böhm, *The Difficulties of Taxonomic Name Extraction and a Solution*, in proceedings of BioNLP, New York, 2006
3. Altova GmbH, www.altova.com
4. <oxygen/>, www.oxygenxml.com
5. IDM Computer Solutions Inc., www.ultraedit.com
6. The OpenNLP project, www.opennlp.org
7. GATE, General Architecture for Text Engineering, gate.ac.uk
8. LingPipe, www.alias-i.com/lingpipe
9. L. Rabiner, B. Juang *An Introduction to Hidden Markov Models*, in IEEE ASSP Magazine, Jan 1986, Volume 3, Issue 1, pp 4-16
10. D. Koning, N. Sarkar, T. Moritz, *TaxonGrab: Extracting Taxonomic Names from Text*, Biodiversity Informatics, 2005
11. D. Agosti, *Encyclopedia of life: should species description equal gene sequence?*, Trends Ecol. Evol. 18: 273, 2003
12. E. Wilson, *The encyclopedia of life*, Trends Ecol. Evol. 18: 77-80, 2003
13. FindIT, names.mbl.edu/tools/recognize
14. Apache Lucene, lucene.apache.org/java/docs
15. D. Butler, *Mashups mix data into global service.* Nature 439: 6-7, 2006
16. E. Gillingham, *Blackwell launches 3000 years of digitized journal backfiles*. Blackwell Publishing Journal News 15, July 2006.