Creating Digital Resources from Legacy Documents – an Experience Report from the Biosystematics Domain

Guido Sautter[1], Klemens Böhm[1], Donat Agosti[2], Christiana Klingenberg[3]

[1] Universität Karlsruhe (TH), Am Fasanengarten 5, 76128 Karlsruhe
{sautter, boehm}@ipd.uka.de
[2] Am. Mus. of Nat. Hist., Central Park West at 79th, New York, NY 10024-5192
[3] Staatliches Museum für Naturkunde, Erbprinzenstr. 13, 76133 Karlsruhe

Abstract. Digitized legacy document marked up with XML can be used in many ways, e.g., to generate RDF statements about the world described. A prerequisite for doing so is that the document markup is of sufficient quality. Since fully automated markup-generation methods cannot ensure this, manual corrections and cleaning are indispensable. In this paper, we report on our experiences from a digitization and markup project for a large corpus of legacy documents from the biosystematics domain, with a focus on the use of modern tools. The markup created covers both document structure and semantic details. In contrast to previous markup projects reported on in literature, our corpus consists of large publications that comprise many different semantic units, and the documents contain OCR noise and layout artifacts. A core insight is that digitization and automated markup on the one hand and manual cleaning and correction on the other hand should be tightly interleaved, and that tools supporting this integration yield a significant improvement.

Keywords: Semantic XML Markup, Digital Resources, RDF Generation

1 Introduction

Documents marked up according to specific XML schemas are important resources in the sciences. They can be stored in databases or can be used for RDF generation, facilitating highly specific search, data mining, and machine reasoning. A prerequisite is that the markup is accurate. Fully automated digitization and markup-generation methods cannot ensure this. This makes (manual) correction indispensable.

The objective of our work has been to make all publications on the ant fauna of Madagascar available as machine-readable resources, marked up as TaxonX documents [4]. The project started in January 2007, and two biologists have worked on it for more than a year. Marking up documents has consumed most of their time, and the goal of this paper is to report on the experiences we have gained with this work. There are 119 documents in total, comprising about 2,500 pages with roughly 1,000,000 words. The most important parts are the more than 4,000 taxonomic treatments, i.e., atomic semantic data units within the documents. The 119 documents are from the last 250 years, with a broad variety in layout and style. The documents

are written in five different languages: English (32 documents), French (54), Italian (5), German (16), and Latin (12). Consequently, reliable rules for automated markup are hard to find/formulate, and automated markup generation requires considerable manual corrections. That high diversity is one reason why our project has been more ambitious than previous ones. Our original idea had been to run automatic digitization and markup tools over the corpus and then correct the result manually. However, we realized soon that this approach is too undifferentiated and not efficient. The reason is the complex structure of the markup and of its generation process, which consists of many steps. Some steps depend on others, i.e., the results of some steps serve as input for later ones. The markup process and correction should be intertwined, i.e., markup generated should typically be corrected right away, before other steps make use of it. Otherwise, errors propagate and add up. This means that we had to deal with the markup process much more extensively than anticipated originally.

The Madagascar corpus has a number of characteristics that have added to the difficulty of markup generation: (1) In the printed originals, the logical reading order of the text is not always identical to the physical one. To improve the search opportunities, it has been necessary to restructure the text to its logical reading order. In particular, individual treatments should not be interleaved with each other or with other parts of a document. (2) The text needs to be free of artifacts originating from print layout, like page numbers. Inclusions like captions and footnotes have to be moved out of paragraphs if they interrupted the text flow otherwise, and hyphenated words need to be rejoined. (3) The individual treatments must be marked up and become accessible individually. (4) To facilitate specific queries and RDF generation, domain-specific details (taxonomic names and locations in this current context) need to be marked up. (5) 'Important' abbreviations, e.g., those within taxonomic names must be resolved to their full meaning. This is because their meaning is clear in the context of the publication as a whole, but not in a treatment looked at in isolation.

Other legacy documents whose digitization would be valuable have similar characteristics. In biomedicine, for instance, historical reports on diseases are valuable to reconstruct the spread of epidemics. Digitization of such documents has to deal with Issues 1, 2, 4, and 5. Historical newspapers and books are important resources as well. Projects creating machine-readable resources from legacy documents will have to deal with most the above Issues 1-5. While various projects aim at digitizing printed information (e.g., the Google Libraries Project), they do not always go beyond automated OCR and spend relatively little effort on error correction. The markup we have added to the Madagascar corpus is hardly within their scope. Further, previous corpus-creation projects which bear some similarity with ours have not faced the same complexity. While projects like [13] have shown that NLP (natural language processing) in combination with subsequent manual correction is viable, their setting has been simpler. A characteristic of [13] and others is that it has dealt with small pieces of text that are individual semantic units (in particular, MEDLINE abstracts [16]) that are clean, i.e., the markup is accurate, and annotations only refer to words within sentences. We in turn deal with large publications comprising many treatments. As an effect of digitization, the documents also contain a lot of optical character recognition (OCR) errors and layout artifacts, which need to be cleaned up.

To support interleaving automated markup generation with manual correction, we have developed the GoldenGATE Editor [18]. It seamlessly integrates both automated

markup components and custom dialogs that support users in correcting the results of automated steps. Further, while users can fully edit textual content, XML editing works on element level. This is helpful in two ways: The user does not have to deal with XML syntax details, i.e., on the character level, and the editor can keep the document well-formed. In this way, it combines the functionality of commercial XML editors like Altova XMLSpy [2] with that of NLP frameworks like GATE [9].
Many insights we have gained with our work are not domain-specific, but are of general interest when creating machine-readable resources from legacy documents: (1) Appropriate ordering of the individual steps may reduce the effort significantly. For instance, instead of first identifying sections and then their titles in a top-down fashion, the reverse order yields higher accuracy. This is because the titles help to identify where sections start. (2) To assist users in correcting auto-generated markup, it is worthwhile to create specialized document views that only provide the required information and options, and render work more intuitive. (3) The effort for manual corrections depends on characteristics of the components that generate the markup, especially when extracting document details: While a user has to search for false negatives by sifting through the whole document, he can sort out false positives rather easily using, for instance, a list view of all extracted details. Hence, it is advantageous to optimize for recall, even at the cost of some precision. (4) A plug-in based architecture like that of the GoldenGATE editor is useful: It facilitates quick deployment of enhanced automated components that follow new insights gained during the work.

Paper outline: Section 2 discusses related work. Section 3 describes the document corpus, Section 4 the software used. Section 5 describes the markup process. Section 6 reviews some statistics and features an outlook. Section 7 concludes.

2 Related Work

Annotated corpora have been created in different domains, to provide input data for document analysis. Projects like the Penn Treebank Project [13] show that NLP reduces the effort. According to publications [5, 8, 12, 13], most of these projects work in two steps: First, NLP tools process the texts. Second, domain experts manually correct the output of the first step, up to a 98% level of inter-annotator agreement in the case of [13]. The annotations[1] cover low levels in the document structure, i.e., part-of-speech and sentence structure [13], word sense [8], named entities [5], and domain-specific terms [12]. However, we are not aware of any studies on how to mark up higher levels of the structure, like sections, treatments, subsections, and paragraphs. This is different from corpus annotation in two ways: First, corpus annotation usually works on small pieces of text that form single semantic units, e.g., the MEDLINE abstracts [16] used for the GENIA corpus [12], and annotations are usually restricted to words and phrases (comparable to the taxon names and locations in our documents, but without the semantic appreciation that comes, for instance, with geo-referencing). In contrast, the documents we work with consist of several (up to 100) treatments, and the markup covers both them and their

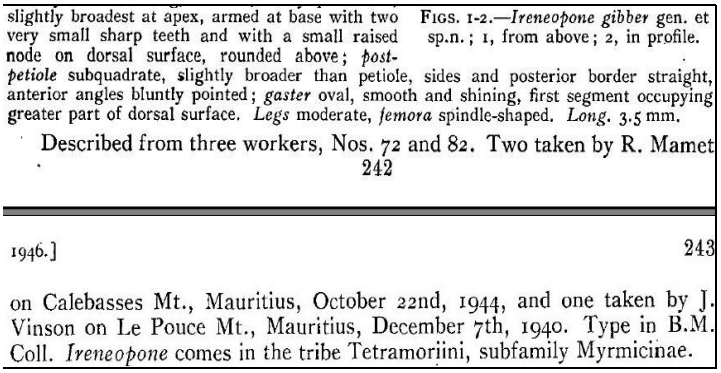
[1] Annotations usually refer to small pieces of text, usually individual words, with no specific representation. As opposed to this, XML markup can also express document structure.

subsections. This results in a much more complex procedure. Second, as opposed to uniform and clean documents like MEDLINE abstracts, our documents are diverse, e.g., different print layouts used over time. Further, they contain layout artifacts like page titles and noise, notably OCR errors, which require extensive cleanup.

3 The Documents

In this section, we describe the documents and the XML markup we have created.

The **Biosystematic Documents** we have worked with are scientific publications from the biosystematics domain. They categorize, describe and discuss living organisms and relationships between them. They have a typical structure: (1) The body contains so-called **treatments**. A treatment is a section referring to one **taxon**. It consists of subsections, each covering one aspect of the taxon (to be explained below and in Section 5.1). A taxon is a node in the tree of life at any level, e.g., a genus or a species. (2) In its **nomenclature** subsection, each treatment provides the reference to the taxon it deals with, usually right at its beginning. (3) Each treatment may comprise a **morphological description** of its taxon and **locations** where specimens have been collected. In addition, treatments may also include subsections covering other aspects. Only markup covering treatment boundaries and structure, taxon names, and locations and morphological descriptions, respectively, allows for creating useful RDF statements from the document content.


slightly broadest at apex, armed at base with two very small sharp teeth and with a small raised node on dorsal surface, rounded above; *post-petiole* subquadrate, slightly broader than petiole, sides and posterior border straight, anterior angles bluntly pointed; *gaster* oval, smooth and shining, first segment occupying greater part of dorsal surface. *Legs* moderate, *femora* spindle-shaped. *Long.* 3.5 mm.

FIGS. 1-2.—*Ireneopone gibber* gen. et sp.n.; 1, from above; 2, in profile.

Described from three workers, Nos. 72 and 82. Two taken by R. Mamet

242

1946.] 243

on Calebasses Mt., Mauritius, October 22nd, 1944, and one taken by J. Vinson on Le Pouce Mt., Mauritius, December 7th, 1940. Type in B.M. Coll. *Ireneopone* comes in the tribe Tetramoriini, subfamily Myrmicinae.

Figure 1. Page break in the middle of a paragraph

The documents from the last 250 years show a wide range of writing styles, layouts, and levels of print quality. With many documents, the logical structure (sections, etc.) is not in line with the layout structure. The print layout is also responsible for artifacts in the document text after digitization. E.g., there are page numbers, page titles, footnotes, or copyright notices. Figure 1 gives an example: A page border (the gray bar), page numbers and a page title interrupt the paragraph "*Described from three workers … the tribe Tetramoriini, subfamily Myrmicinae.*". In addition, there may be hyphenations from the print layout, which are in the way of NLP.

For the **Output XML Documents** to yield maximum benefit, they should be marked up according to TaxonX [4], a dedicated XML schema for biosystematics. It

provides markup for several levels of document structure, like sections or treatments, and for specifics like taxon names or locations, together with normalizations, like geo-coordinates. TaxonX imports the DarwinCore [7] schema – a de-facto standard in biosystematics – for the normalized data. TaxonX markup facilitates many usages of the documents. A reasoner can compute the dispersal of a taxon (i.e., all locations where it occurs), the fauna of a location (i.e., all taxa that occur at a given location), or the taxon a specimen belongs to, based on observable morphological features.

4 Tools Used

For the initial Optical Character Recognition process (OCR), we used Versions 8 and 9 of **ABBYY FineReader** [1], a commercial scanning and OCR tool, with a specific setup: Since our source documents are written in different languages, sometimes within one document, we use the multiple languages mode. The program then considers more than one language for the spell check. We have integrated two domain-specific dictionaries we have built ourselves, containing parts of known taxon names and technical terms occurring in morphological descriptions. For text recognition, ABBYY FineReader can deploy user patterns, i.e., trained configurations for a given layout style or font. They help processing old documents with uncommon layout where the built-in type recognition would fail otherwise. We have created training files for 15 journals that contain some of the articles.

To support users correcting markup as far as possible, we have designed and implemented the **GoldenGATE Editor**[2] [18]. It tightly integrates automated markup generation and assistance features for subsequent manual correction. To facilitate deployment of tools for automated markup generation, the editor provides extension interfaces for their integration. We will refer to such components integrated through one of these interfaces as plug-ins. In addition, the editor provides high-level assistance functions for manual editing: While users can edit textual content as usual, XML editing works on element level. This is helpful in two ways: The user does not have to deal with XML syntax details (i.e., on the character level), and the editor can always keep the document well-formed. This is helpful for users who are domain experts like biologists rather than XML professionals. A laboratory experiment [19] has shown that the combination of assisted manual editing and NLP-based markup generation reduces the markup effort significantly, compared to commercial XML editors. In the experiment, the users worked with small documents with markup according to a simple XML schema. This paper in turn will show that the functionality described in this paragraph and in [18] is even more useful in a real-world setting.

As an example of a plug-in we have developed for the GoldenGATE editor, Figure 2 shows the **Annotation Selector**. It displays XML elements as a list to provide a view of some of the document content. An XPath expression selects the elements to display. The user can edit and remove each element. This view is helpful to correct auto-generated detail markup, in particular to sort out false positives.

[2] Available for free download and use at http://idaho.ipd.uka.de/GoldenGATE/

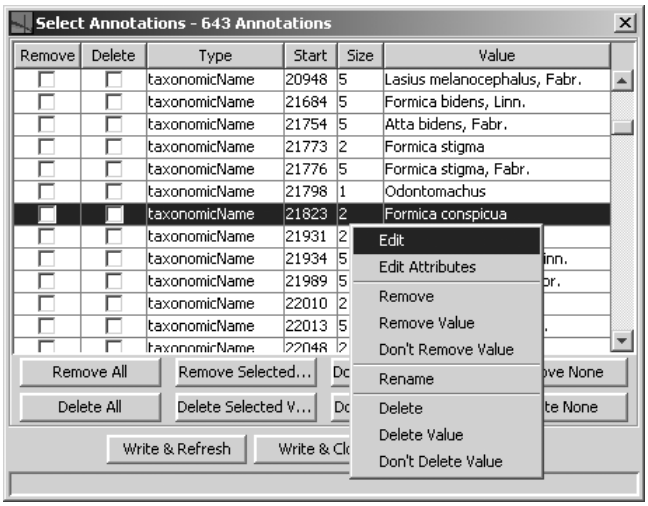


Figure 2. List view of XML elements for correcting detail markup

For a detailed description of the GoldenGATE Editor please see [18] or the documentation at http://idaho.ipd.uka.de/GoldenGATE/.

5 Document Markup

In this section, we describe the automated steps to mark up the documents, and how we have interleaved them with manual correction to prevent error propagation. Many steps are generic, i.e., are necessary whenever creating digital resources from digitized legacy documents, e.g., artifact removal and normalization. Others are specific to our project to some degree, but similar steps are likely to be part of projects in other domains as well. We also discuss how arranging the steps in an appropriate order can minimize errors and thus the correction effort.

5.1 The Markup Steps

We now describe the digitization and markup steps, grouped as follows: OCR & normalization; structural markup; and detail markup. This is not the actual order of the steps, but eases the presentation. We will mention various automated markup-generation tools in this section. Most of them are plug-ins to the GoldenGATE editor. Unless stated otherwise, we have implemented them ourselves. Paragraphs labeled *INSIGHTS* contain some generalizations which others might find helpful.

5.1.1 OCR Artifact Removal & Normalization

The step for OCR, error correction, and normalization is not specific to our project. It is an endemic part of digitizing and marking up any corpus of legacy documents.

Scanning & OCR. ABBYY FineReader scans and transforms the printed legacy documents to character data via OCR. This step produces an HTML document, the starting point for further processing with GoldenGATE. We use the following output options of ABBYY FineReader: (a) *Remove all formatting.* The text quality in older publications sometimes is so bad that fonts and style may wrongly change several times within a word. (b) *Simple (compatible with all browsers).* (c) *Keep line breaks.* Line breaks have to be retained initially, because they help to detect and re-join hyphenated words. (d) *Use solid line as page break (keep page breaks).* Likewise, we initially keep page breaks, to identify certain artifacts like page titles, page numbers, and footnotes, which usually are next to page breaks. (e) *Do not save with images.* Since the XML documents are intended as XML resources, we do not keep images. However, the URLs of images can be stored as attributes of their respective captions.

Layout-Artifact Detection. To prevent errors in later steps, we have to make sure that the structure of the text is intact. In printed text, paragraphs can be broken, for two reasons: (1) They can run over page breaks, and page number and page title interrupt the text flow. There can also be footnotes prior to the page break. (2) Paragraphs may have figures or tables with captions embedded, which also interrupt the text flow. To recognize and remove these insertions, a dedicated tool first marks up the pages, from one page break to the next one. Other tools then search for page titles, page numbers etc. at the top and at the bottom of the pages. One of them turns the page numbers into attributes of the paragraphs of the page to facilitate page-level citation.

Paragraph-Border Correction. Paragraph borders are sometimes faulty in OCR output, due to varying indentation and spacing schemes. I.e., a paragraph may be split in the middle, or two paragraphs are marked up as one. Respective checks can be automated to some degree, e.g., based on capitalization or the punctuation. But these checks can only point the user to likely errors, since the evidence tends to be too incomplete and unreliable for automated correction. Before removing recognized artifacts, like page titles and footnotes, we manually check which paragraphs are artifacts and which are not. A specialized document view supports this, displaying the paragraphs of one page at a time for better overview.

Paragraph Normalization. Once the paragraph boundaries are in place, a respective plug-in removes line breaks from within the paragraphs and re-joins hyphenated words, e.g., ‘fau-<line break>na’ becomes ‘fauna’. It also removes the page markup, which is not needed any longer.

MODS Referencing. To ease citation and provide a unique identifier for the XML document, we import respective metadata [14], using the MODS data format[3]. A plug-in obtains the metadata from a web service[4] and inserts it into the document.

Structural Normalization. The next step is to remove the layout artifacts. Deleting page titles is unproblematic because they do not contain useful information. Footnotes and captions are harder to deal with because they represent content. Thus, if they do not interrupt the text flow of another paragraph, they stay in place. Otherwise, a dedicated plug-in moves them to the position behind the paragraph which they interrupt. This changes the structure of the text, but not significantly. Further,

[3] Metadata Object Description Schema

[4] http://atbi.biosci.ohio-state.edu:210/hymenoptera/hym_utilities.format_ref?style=MODS&id=<documentIdNumber>

footnotes are usually printed right before a page break, independent of the exact position in the text referencing them. Captions in turn stand next to the figures they belong to, and the position of the latter typically depends on layout considerations. Thus, moving footnotes and captions out of paragraphs is unproblematic.

INSIGHTS. *(a) Since correcting OCR errors causes considerable effort, OCR quality should be favored over full automation and throughput if markup generation is an objective of the project. For mass digitization in turn, where markup generation might only be an option later on, we recommend providing a link from the OCR-ed documents to the original page images. These images have turned out to be helpful for later corrections. (b) Even though marking up the layout structure of the printed document creates additional artifacts, having this markup temporarily is helpful for cleaning. In particular, marked up pages are reliable evidence for finding other artifacts, e.g., page title or footnotes.*

5.1.2 Structural Markup

The markup we create introduces two levels of document structure, namely the treatments and their inner structure.

Treatment Markup. Once the document is structurally clean, a respective plug-in marks up the treatments. Like sections in other documents, a treatment consists of one or several paragraphs. I.e., the plug-in uses rules to group the paragraphs into treatments and sort out the ones that do not belong to any treatment. To simplify correction of the treatment boundaries, we again use a specialized document view: The user can visually check if the paragraphs are correctly grouped into treatments and can correct errors.

Table 1. Types of subsections in treatments

Type	Primary Content
Description	Morphological description of a taxon
Materials Examined	Locations where the specimens described in Description have been collected
Biology	Behavior of taxon, interactions with environment, nesting preferences
Diagnosis	Explanation of differences of taxon to related taxons, plus identifying features.
Discussion	Explanation why these differences justify making them a taxon of their own
Distribution	Summary and distribution pattern of a taxon (e.g., 'throughout southern Africa'), based on material presented in Materials Examined
Etymology	Etymology (origin) of the taxon name

Structure of Treatments. A treatment usually consists of several subsections: The nomenclature subsection always exists and provides the name of the taxon the treatment refers to. Other subsection types do not always exist and need to be marked up if they do (cf. Table 1). As the treatments, their subsections also consist of one or several paragraphs each. A rule-based plug-in groups the paragraphs of each treatment into subsections. To make manual corrections as simple as possible, we use the same approach as for the treatment boundaries: In a dedicated document view, the user can visually correct the grouping of the paragraphs into subsections.

INSIGHTS. *Elements of coarser structure, e.g., treatments, typically are groups of paragraphs. It turns out that marking up the paragraphs before the coarser elements is favorable over a top-down approach. This is because, given the paragraphs, one can mark up the coarser elements by grouping paragraphs relatively easily. Respective document views have turned out to be very helpful to correct the groupings.*

5.1.3 Detail Markup

The markup we create comprises two types of semantic details, namely taxon names and locations. Both are normalized, to facilitate search, data mining, and creating RDF statements. In addition, we add identifier numbers to the former and geographical coordinates to the latter. While the detail markup generated is specific to our context, the general problem of marking up important details is not: In the context of historical medical documents, for instance, geographic details are essential to compute the spread of an epidemic, as are bacteria names.

Taxon-Name Markup. The taxon names are the most important details in the documents. To detect and mark them up automatically, we use a dedicated algorithm [17], implemented in a plug-in. It is both rule- and dictionary-based, using regular expression patterns for the rules. In addition, it extends the lexica dynamically as it finds new taxonomic names in the documents. Further, it can be tuned towards precision or recall. A list view of the taxonomic names avoids sifting through the document for corrections. Since this view only helps with false positives, we tuned the component towards maximized recall to avoid false negatives.

Since older publications in particular show a variety of layout styles, including capitalization and punctuation schemes for taxonomic names, we had to adjust the rules frequently in the beginning. To simplify adaptation to new styles, we ended up implementing a specialized editor for the regular expression patterns of the rules.

The next step is to extract the normalized form of the taxon names, to facilitate exact searching and simplify RDF generation. In particular, this means adding the 'meaningful' parts of the name as attributes to the XML element marking up the name. The meaningful parts are the ones that carry taxonomic information, as opposed to those clarifying name authority or labeling meaningful parts. For example, in the taxonomic name *Drosophila melanogaster* Meigen, *Drosophila* is the genus, *melanogaster* the species, and Meigen the author of the species name *melanogaster* (the author of the genus name *Drosophila* is Fallén, which is not given here). For search and data mining purposes, we are only interested in the meaningful parts *Drosophila* and *melanogaster*, i.e., the parts that specify the position of a taxon in the tree of life. We add these parts as *genus* and *species* attributes to the XML element marking up the taxon name. We do not make an attribute of Meigen.

A plug-in parses the meaningful parts from the taxon names as they occur in the document. Since some parts of taxon names may be abbreviated or omitted completely, the next step is to fill in these gaps, based on other taxon names in the document. *Drosophila melanogaster*, for instance, may be referred to as *D. melanogaster* later in the same document. In the document as a whole, the meaning of the abbreviation *D.* is clear, but this may not hold for a treatment in isolation. We therefore resolve the abbreviation to *Drosophila*. Similarly, co-reference analysis in any other domain may resolve pronouns and last names to a full person name mentioned earlier in a

document. So even without much context information, it is clear to whom a sentence refers to. To correct a normalization result, we use a dedicated dialog that allows for selecting the full form of an abbreviated part from a respective list. In addition, users can change the meaningful parts of the taxon names (i.e., the parts of the name that carry taxonomic semantics, see example above). A parser then automatically adjusts the meaning of the other parts to comply with the correction. Throughout our work, we have kept improving this dialog based on feedback from the users. The improvements include displaying the page number where the taxon name occurs in the document, and displaying the context of the taxon name in the document, about 10 words to the left and right. Both measures simplify deciding which part of the taxon name has which meaning. In its current status, the dialog reduces the effort for the normalization of taxon names by 95% according to our estimations; see Section 6.

After the meaning of all taxon names is clear (i.e., there are no unresolved abbreviations), the next step links these names to an existing biosystematics database. In particular, it obtains their Life Science Identifiers (LSIDs, [3]) from the Hymenoptera Name Server [11]. LSIDs allow assembling very precise RDF statements on a specific taxon. In addition, they facilitate integrating the treatments into mashups, i.e., brief compositions of all data available on a given taxon from various sources. If a taxon name is not yet in the database of that name server, i.e., there exists no LSID for it, we upload it to the server and obtain a newly generated LSID in return.

Location Markup. Another important piece of information in the documents is the locations. We have implemented a combination of named entity recognition techniques [6, 15] to mark them up automatically. Cleaning location markup is similar to the one of taxon names, i.e., using the list view to sort out false positives.

To give the locations a unified unambiguous representation, a respective plug-in adds attributes with their geographical longitude and latitude. It obtains this information from the GeoNames web service [10]. For ambiguous location names, the user can select the correct coordinates from a respective dialog.

INSIGHTS. *(a) Marking up important details is a common step when creating digital resources from legacy documents, since the details are essential to extract RDF statements. The high diversity of legacy literature, however, poses new challenges: For instance, to reduce the correction effort, we frequently had to adjust the rules for taxonomic name extraction. The situation is similar when patterns are used for detail extraction, due to the numerous variants of spelling, capitalization and punctuation in legacy literature, which are impossible to foresee. The pattern editor has turned out to be helpful, and we would also expect this for other projects. (b) An important optimization goal should be maximizing recall, even at the cost of (some) precision, as opposed to f-score or accuracy, which are relevant criteria in NLP research. Namely, sorting out false positives from a list view of marked up details has turned out to cause much less effort than searching the document for false negatives.*

5.2 Using Generic Markup

Instead of creating TaxonX markup directly, the preprocessors produce and work with generic ‘out-of-schema’ XML elements. Only the last preprocessor transforms the documents to TaxonX using XSLT. We have chosen this approach for several reasons:

(1) **Temporary Markup.** Some markup steps require XML elements that are temporary. The detection of layout artifacts, for instance, makes use of markup identifying the pages of the printed document. Since the pages are not part of the final markup, TaxonX does not provide an element for them. Thus, we have to use 'out-of-schema' XML elements, and the working documents would not be valid TaxonX anyhow. (2) **Target-Schema Evolution.** In order to react to various unforeseeable special cases we encountered in the project, e.g., whole treatments written in captions or footnotes, we have proposed significant refinements and modifications of the TaxonX schema itself. Generic markup allows dealing with those cases right away (i.e., without having to initiate schema modifications and to wait for the outcome) without encountering validity problems. Generic markup is also helpful with cases which the target schema cannot represent, e.g., a nesting of structural markup elements that is not valid. (3) **Storing Additional Information.** When marking up the documents, we obtain additional useful information, e.g., normalized forms of the taxon names and geo-coordinates of the locations. TaxonX stores such extra data as textual content under the tax:xmldata element, as imposed by DarwinCore [7]. Thus, creating the target markup right away would mix up artifact elements with document text. This would be in the way of further markup steps, especially NLP-based ones. To avoid this, we use generic markup that allows storing the extra data as attributes. (4) **Reusability of Automation Components.** Implementing specialized document views to correct the results of the individual steps has been a significant effort, though – in the long term – this effort is much lower than doing the corrections manually in a generic view. To facilitate reuse of these views, it is helpful not to develop them for a specific XML schema. (5) **Integration of Third Party Components.** Using existing NLP components for automated markup generation becomes harder when producing markup that must conform to a schema. This is because then, the output of the component has to be transformed right away, to comply with the schema.

Most of these advantages also hold for other projects which mark up legacy documents. The only item specific to our project at first sight is the representation of the taxon names in TaxonX. However, final markup generated by an intermediate step may be in the way of subsequent processing steps. Thus, the distinction between "working markup" and final markup should be helpful in other domains as well.

INSIGHTS. *Working with generic markup and transforming it into the target schema only at the end is advantageous: Compared to schema-bound markup, one is more flexible regarding the deployment of NLP components, the explicit representation of extra information, and structural variations it can represent.*

5.3 Order of Markup Steps

Performing the markup steps in top-down order, i.e., first structure, then details, results in unnecessary effort. In particular, details are reliable evidence to avoid errors when marking up the document structure. This leads to an alternative order of the steps (see Figure 3). We first mark up the details, and then use these details as evidence when generating the structural markup.

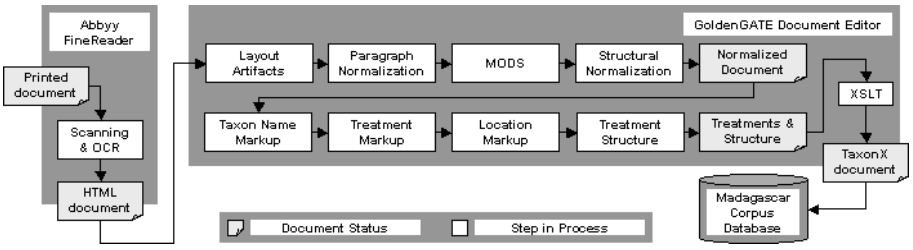


Figure 3. The markup & cleaning process as we use it

Treatment Markup. We have observed that almost every treatment starts with its nomenclature subsection. This subsection usually contains one rather short paragraph and starts with a taxon name. This means that the taxon names reliably identify nomenclature paragraphs. Based on these paragraphs, a respective plug-in generates the treatment markup, assuming that a treatment starts at the start of a nomenclature paragraph and ends immediately before the next nomenclature paragraph. However, this assumption does not always hold: There can be document text between treatments, and there usually is text after the last treatment, namely acknowledgements and the references sections. Thus, we cannot eliminate manual correction altogether, but we can reduce the effort considerably.

Internal Structure of Treatments. We observe that certain details identify some subsections of treatments rather clearly: (1) A series of taxon names identifies a discussion or diagnosis subsection. (2) The presence of location names indicates a distribution or 'Materials Examined' subsection. (3) The presence of the names of body parts (observable morphological features of specimens) indicates a description, diagnosis, or discussion subsection. – Further, we have observed that 'discussion' subsections are much more frequent than 'diagnosis' subsections, and 'Materials Examined' subsections are frequent than subsections with plain distribution data. With these observations, we have been able to formulate more reliable rules for the component that groups the paragraphs of a treatment into subsections.

INSIGHTS. *(a) When several markup steps need to be done, yielding markup of different granularities, a good ordering of the steps can facilitate a higher degree of automation and prevent errors, compared to the top-down approach. (b) Our approach for treatment markup is applicable in other settings as well. In general, details are reliable evidence to identify the document structure. For instance, once section titles are marked up, one can mark up sections automatically, with few errors. (c) Classification rules similar to the ones used here for the inner structure of treatments can avoid errors when marking up the document structure: For instance, in scientific publications regardless of the domain, the presence of person (author) names, titles, and years is a reliable hint that the section is a References section.*

6 Statistics

One of the biologists of our project has marked up about 100 documents; Figure 4 graphs the average working time per page. When our work started, both the GoldenGATE editor and the markup generation plug-ins were less sophisticated than they

currently are (cf. Subsection 5); marking up one document page took over 12 minutes. Over time, based on our experiences, we improved the editor, the markup-generation plug-ins, and the specialized document views. The average working time decreased to about 2 minutes per page, a reduction by over 80%. The outliers are documents with large pages or with very poor OCR quality. The biologist who did the markup accounts about 3 to 4 minutes of the 10 minute speedup per page to training effects. The remaining 6 to 7 minutes result from the refinement of the tools. Unfortunately, we could not validate these numbers more rigidly without actually disturbing the users in their work. This is because this project took place under regular working conditions, with real output, not as a laboratory experiment. The user, according to her own assessment, was familiar with the tools after about 10 to 15 documents, i.e., about 100 pages. The average working time per page was still around 9 to 10 minutes at that point. Thus, this might serve as further evidence that the remaining speedup is largely due to technical improvements.

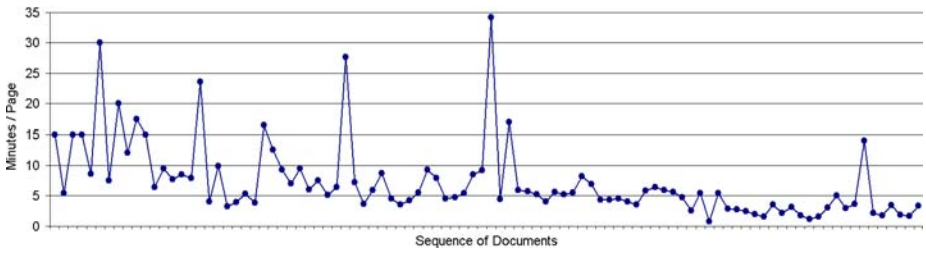


Figure 4. Average working time per page

Table 2. Time per step (in minutes) for 10 pages

Markup Step	Time (With Automation)	Time (Manually)
Scanning & OCR	**38:00**	
Layout Artifact Detection	3:16	6:25
Paragraph Normalization	0:10	2:39
Get MODS Metadata	0:43	
Taxon Name Markup	1:41	8:21
Taxon Name Normalization	0:45	20:10
LSID Referencing	3:53	14:40
Treatment Markup	1:02	2:32
Location Markup	1:40	2:11
Treatment Structure Markup	3:20	17:30
Structural Normalization	0:05	2:10
Markup Total	**16:35**	**77:21**
Total	**54:35**	**115:12**

Table 2 graphs the amount of time users spent with correcting the markup from the different steps. Correcting the document structure takes considerable time. This is because the OCR errors do not follow any regularity, at least according to our perception. Consequently, users have to clean up every page. Getting the LSIDs for the taxon names takes long due to the web-service lookups. Cleaning the inner structure

of treatments takes rather long as well. This is because the respective classification technique still yields many errors. Despite this, the effort with the respective document view (3:20) is still less than 20% of what it would be without it (17:30).

Outlook. Our work on the Madagascar corpus[5] is now finished. While this corpus allows biosystematicists to experiment with new methods (like machine reasoning and data mining) and showcases the benefit of converting legacy documents into digital resources[6], it comprises only a small fraction (< 1%) of the available literature. We plan to investigate how to perform this conversion more efficiently: (a) To improve document normalization, we plan to analyze the errors the automated components have produced on the Madagascar corpus more systematically. To further improve the markup of treatment subsections, we plan to refine the respective rules, e.g., by multiple levels of automated classification, and by more extensive use of keywords. (b) Despite the high level of assistance and the low error rate achieved in this project, a small group of domain experts cannot take on 100,000 or more pages of legacy documents. We plan to investigate how to distribute the work more efficiently and get more individuals involved.

7 Conclusions

Converting legacy documents into digital resources for semantic web applications is a complex task. Running the complete markup procedure and correcting the final result has turned out to be inefficient. This is because the generation of complex markup involves many steps, and some steps use the output of previous ones as input. Automated processing is rarely 100% accurate, and dependencies between the steps result in error propagation and thus induce high correction effort. Interleaving markup generation and manual correction prevents error from propagating. To support the users, we have developed the GoldenGATE Editor. It tightly integrates automated markup-generation components with support for manual correction, i.e., integrates the basic editing functionality found in conventional XML editors with the markup generation facilities of NLP frameworks. In this paper, we have reported on experiences gained from converting a large corpus of digitized legacy literature into semantic digital resources. The work took place under regular working conditions, not as a laboratory experiment. Biologists from Australia, Brazil, Germany, Taiwan, and the USA have expressed interest in our tools. On the other hand, while the domain currently is systematic biology, our insights should be of interest to other fields as well.

---

[5] Corpus available at http://antbase.org/databases/madagascar.htm

[6] You can try out search functionality based on marked up semantic details at http://plazi.org:8080/GgSRS/search - type “Probolomyrmex tani” in the search field labeled “Name” for a start. The “GoogleMaps” link in the result shows one of the possible applications of detail markup: It plots a map from marked and geo-referenced locations, visualizing the distribution of the ant species *Probolomyrmex tani*.

References

1 Abbyy FineReader, http://www.abbyy.com/finereader_ocr/
2 Altova GmbH, http://www.altova.com
3 Brazma, A., Krestyaninova, M., Sarkans, U. Standards for systems biology. Nature Reviews Genetics 7, pp. 593–605, 2006.
4 Catapano, T. et al. TaxonX: A Lightweight and Flexible XML Schema for Mark-up of Taxonomic Treatments. In Proceedings of Annual Meeting of Taxonomic Data Working Group 2006, St. Louis, MO, USA, 2006.
5 Chinchor, N. MUC-7 Named Entity Task definition. In Proceedings of Message Understanding Conference, Washington, DC, USA, 1997.
6 Cucerzan, S., Yarowsky, D. Language Independent Named Entity Recognition Combining Morphological and Contextual Evidence. In Proceedings of the Joint SIGDAT Conference on EMNLP and VLC, University of Maryland, College Park, MD, USA, 1999.
7 DarwinCore2, http://darwincore.calacademy.org/
8 Fellbaum, C., editor. WordNet: An Electronic Lexical Database. MIT Press, Cambridge, MA, USA, 1998.
9 General Architecture for Text Engineering, http://gate.ac.uk
10 GeoNames, http://www.geonames.org
11 Johnson, N. F. The Hymenoptera Name Server. Home Page: http://atbi.biosci.ohiostate.edu:210/hymenoptera/nomenclator
12 Kim, J.-D., Ohta, T., Tateisi, Y., Tsujii, J. GENIA corpus – a semantically annotated corpus for bio-text-mining. Bioinformatics, pp. i180-i182, Oxford University Press, 2003.
13 Marcus, M. P., Santorini, B., Marcinkiewicz, M. A. Building a Large Annotated Corpus of English: The Penn Treebank. Computational Linguistics, Vol. 19, No. 2, pp. 313-330, 1994.
14 Metadata Object Description Schema. http://www.loc.gov/standards/mods/
15 Mikheev, A., M. Moens, C. Grover. Named Entity Recognition without Gazetteers. In Proceedings of Annual Meeting of European Association Computational Linguistics, Bergen, Norway, 1999.
16 National Library of Medicine: MEDLINE (factsheet: http://www.nlm.nih.gov/pubs/factsheets/medline.html)
17 Sautter, G., Böhm, K., Agosti, D. A combining approach to find all taxon names (FAT). In Biodiversity Informatics, Vol. 3, [Online: https://journals.ku.edu/index.php/jbi/index], 2006.
18 Sautter, G., Agosti, D., Böhm, K. Semi-automated XML Markup of Biosystematics Legacy Literature with the GoldenGATE Editor. In Proceedings of Pacific Symposium on Biocomputing, Weilea, HI, USA, 2007.
19 Sautter, G., Böhm, K., Padberg, F., Tichy, W. Empirical Evaluation of Semi-Automated XML Annotation of Text Documents with the GoldenGATE Editor. In Proceedings of European Conference on Research and Advances in Digital Libraries, Budapest, Hungary, 2007.