Extending Data Models by Declaratively Specifying Contextual Knowledge

Tobias Gradl
Media Informatics Group
University of Bamberg
96047 Bamberg, Germany
tobias.gradl@uni-bamberg.de

Andreas Henrich
Media Informatics Group
University of Bamberg
96047 Bamberg, Germany
andreas.henrich@uni-bamberg.de

ABSTRACT

The research data landscape of the arts and humanities is characterized by a high degree of heterogeneity. To improve interoperability, recent initiatives and research infrastructures are encouraging the use of standards and best practices. However, custom data models are often considered necessary to exactly reflect the requirements of a particular collection or research project.

To address the needs of scholars in the arts and humanities for a composition of research data irrespective of the degree of structuredness and standardization, we propose a concept on the basis of formal languages, which facilitates declarative data modeling by respective domain experts. By identifying and defining grammatical patterns and deriving transformation functions, the structure of data is generated or extended in accordance with the particular context and needs of the domain.

Keywords

Digital humanities; descriptive data modeling; language applications; DARIAH

1. INTRODUCTION

Digital collections of the arts and humanities—like the traditional forms of museums, archives or libraries—provide access to various types of research objects. As either originally digital or digitalized resources, they can be preserved and—if tolerant licenses are chosen and technical infrastructure is available—provided to a greater and potentially distributed public. If digital objects are encapsulated or referenced by metadata, the application of quantitative methods to support qualitative research is further facilitated. Despite the ongoing trend towards the development of standards and best practices for the digitization and description of research data within the arts and humanities, recent studies indicate a hesitant or practically non-existent adaption of standards–other than that of simple Dublin Core (DC) [6, 7]. As a result of the instantiated research infrastructures such as Europeana, DARIAH or CLARIN, standards might increasingly be favored over the definition of custom data models. However, the use and publication of data conforming to custom or legacy data models can be expected to persist as (1) the funding required to restructure data in the collections might not be available, (2) information might expected to be lost when transforming existing data or (3)—even for new digitization projects—a standard that exactly matches the demands might not exist and the collection choses a custom design.

In this paper we present an approach that allows experts within the digital humanities to declaratively model original data—irrespective of the degree of structuredness and standardization. The novelty of our approach consists in the language theoretical foundation of this modeling task, which results in data description facilities that are (1) expressive enough to incorporate the complex models required for scholarly research and (2) reduce technical overhead—allowing domain experts to focus on the semantic aspects of data modeling.

This paper is structured as follows: In section 2 we will introduce two examples that illustrate the types of data our approach is focused on. Section 3 will present a formal foundation of data models into which *labeling functions*—the central element of our approach—are incorporated. After detailing the composition of such functions and providing an overview over the behavior of the implemented framework, we conclude this paper with section 4 and a brief reflection of a research-oriented application that has been implemented on the basis of our framework.

2. CONTEXT

Ultimately, our goal is to provide capabilities for domain experts of the arts and humanities to describe data within its context. By explicating their contextual knowledge, they are enabled to extend and enrich original data. Due to the diversity of the domain, its research questions and data, we developed a concept that abstracts semantic aspects of data modeling from technical problems [1]. The main intention is to allow experts within a specific discipline or collection to focus on those aspects of data processing, which require their domain expertise. Technical problems of data access, decoding, processing or integration are solved in a generic, reusable fashion.

Structured data. As an example of data that is often provided by digital collections, consider the excerpt of a simple DC metadata record taken from the PANGAEA (Data Publisher for Earth & Environmental Science)[1] database. With respect to *atomicity*, the structural decomposition of each element seems obvious and would enhance the structure of the document. Although metadata facilitates access to information, the described digital resource might be particularly relevant for answering research questions. Our concept

[1] https://pangaea.de/

is intended to be applicable for modeling structural elements within unstructured text—allowing the domain-driven enrichment of such data.

```
<oai_dc:dc>
  <dc:creator>Grobe, Hannes</dc:creator>
  <dc:date>1996-02-29</dc:date>
  <dc:format>text/tab-separated-values, 1148 data points<
      /dc:format>
  <dc:language>en</dc:language>
  <dc:coverage>LATITUDE: 68.556667 * LONGITUDE:
      -21.210000 * DATE/TIME START: 1994-09-19T14:56:00
      * DATE/TIME END: 1994-09-19T14:56:00 * MINIMUM
      DEPTH, sediment/rock: 0.0 m * MAXIMUM DEPTH,
      sediment/rock: 11.5 m</dc:coverage>
  <dc:subject>ARK-X/2; AWI_Paleo; Denmark Strait; Gravity
       corer (Kiel type); Ice rafted debris; IRD-
      Counting (Grobe, 1987); Paleoenvironmental
      Reconstructions from Marine Sediments @ AWI;
      Polarstern; PS2646-5; PS31; PS31/162</dc:subject>
</oai_dc:dc>
```

Listing 1: Pangaea DC example

Unstructured data. Listing 2 shows the first few lines of the wikipedia article on Gustave le Bon. The document is composed of a mixture of structured and unstructured elements as the document itself conforms to the Extensible Markup Language (XML) and the MediaWiki export schema and the content of the *text* element conforms to Wiki markup. As such, the unstructured text of the article is complemented with structural components such as headings and links.

```
<page xmlns="http://www.medi...export-0.9/">
  <title>Gustave Le Bon</title>
  <ns>0</ns>
  <id>104619</id>
  <revision>
    <id>135522322</id>
    <parentid>135491542</parentid>
    <timestamp>2014-11-04T17:40:26Z</timestamp>
    <contributor>
      <ip>146.52.78.48</ip>
    </contributor>
    <comment>/* Der Rassebegriff bei Le Bon */</comment>
    <text xml:space="preserve">[[Datei:Gustave Le Bon.jpg
        |thumb|Gustave Le Bon im [[fin de siècle]]]] '''
        Gustave Le Bon''' (* [[7. Mai]] [[1841]] in [[
        Nogent-le-Rotrou]]; âĂă [[15. Dezember]]
        [[1931]] in [[Paris]]) gilt als Begründer der [[
        Massenpsychologie]]. Seine Wirkung auf die
        Nachwelt, wissenschaftlich auf [[Sigmund Freud]]
         und [[Max Weber]], politisch insbesondere auf
        den [[Nationalsozialismus]] und seine ...
```

Listing 2: Wikipedia example

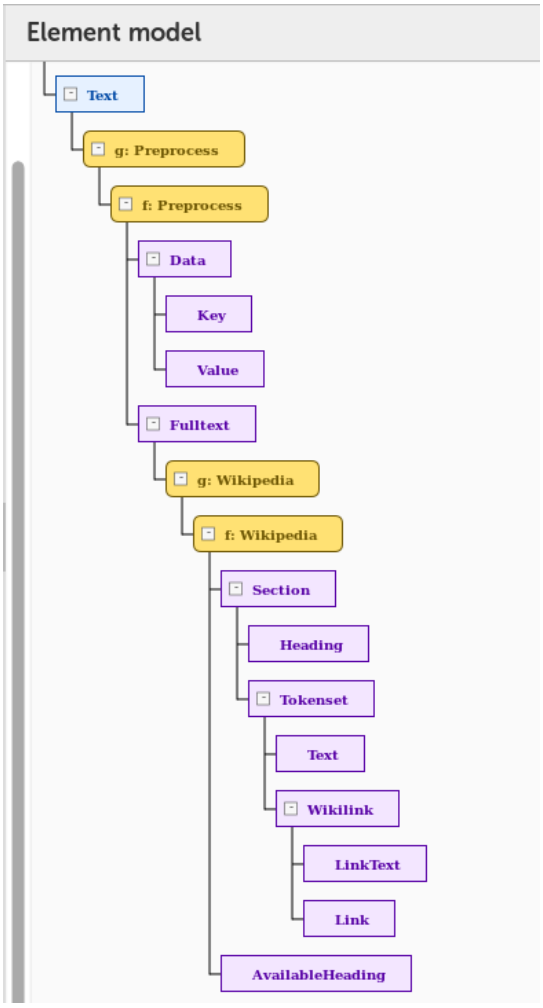

Figure 1: Modeled tree of the wikipedia example

3. DATA MODELING

3.1 Perspectives on data

Based on the foundation in [8] and [3], semi-structured data models can be interpreted in terms of finite structures $\langle N, T, R, P \rangle$—regular-tree grammars with the finite sets of nonterminals (N) and terminals (T), the root symbol ($R \in N$) and the set of production rules (P). This definition allows the production of a semi-structured document according to its e.g. XML or JavaScript Object Notation (JSON) schema formulated constraints. As such, we consider the interpretation of a schema as $\langle N, T, R, P \rangle$ the *parsing-oriented perspective* on semi-structured data. The definition allows the specification of production rules of the form $n \rightarrow te_c$, where $n \in N$, $t \in T$ and $e_c \subset N$ reflects the content model that is defined over the set of non-terminals.

With regard to the introduced examples, this definition allows the production of terminal symbols of the XML documents from nonterminals and hence the formal definition of an XML schema. Although a strictly parsing-oriented perspective is necessary for the syntactically correct interpretation of data, the presented definition does not allow the extension of document structure based on the content of the elements.

To allow the representation of substructures or alternative elements within the formal definition of a schema, we extend the parsing-oriented view to a 6-tuple $S = \langle N, T, R, P, L, F \rangle$, where N, T, R and P form components of the original definition. The components of L and F provide the extension of the schema, where:

- L forms a set of labels and
- F is a set of labeling functions $x \rightarrow le_l$, where:
  - $x \in (N \cup L)$,
  - $l \subseteq L$ and
  - $e_l := \{I, op\}$ defining a function over a set of input values $I \subseteq N$ and an operation of the arity $|I|$.

Figure 1 shows the editor component of the modeling interface of our framework[2]. In this editor, the blue nodes represent the nonterminals. The yellow nodes are labeling functions, which are formed of a grammatical and a transformation component. The purple nodes finally represent produced labels. Through the hierarchy, parenting nodes of labeling functions also define the set of input values.

3.2 Labeling functions

Within the framework, labeling functions are composed of the two constructs *grammars* and *functions*, which represent two distinguishable modeling tasks. Whereas grammars are used to define the grammatical constraints that generate a language—i.e. the Domain Specific Language (DSL) that an element conforms to—functions build on resulting syntax trees to transform data into subsequent labels.

The example in figure 1 shows an intermediary result of modeling the presented Wikipedia example: the *grammar* `g: Preprocess` for the separation of encapsulated structured information from unstructured text is inserted below the `Text` element. The *transformation function* `f: Preprocess` applies commands to produce key/value pairs of structured data and the remaining article (`Fulltext`). An additional grammar `g: Wikipedia` then decomposes Wiki markup from encapsulated textual content. The transformation function `f: Wikipedia` is then applied on the produced parse tree and generates the element hierarchy modeled under `Section`.

Description grammars

The specification of data with respect to syntactical and semantic constraints is accomplished by means of an individual, element-specific DSL [5].

```
page       : container+;
container  : (block | preface) block*;
preface    : content;
block      : h1block | ... | h6block;

h1block    : h1  (h2block | ... | h6block | content)*;
...
h6block    : h6  (content)*;

h1         : H1_OPEN title EQ;
...
h6         : H6_OPEN title H6_CLOSE;

content    : (tokenset | categoryContainer)+;
title      : tokenset+;
tokenset   : text
           | link;
...
text       : TEXT | EXCL | EQ;

intLink     : intLinkOpen intLinkCont INT_LINK_CLOSE;
intLinkCont : intLinkComp? linkValue;
intLinkComp : intLinkRes INT_LINK_SEP
              (linkContent INT_LINK_SEP)*;
intLinkRes  : linkContent;
...
```

Listing 3: Wikipedia grammar

Listing 3 shows an excerpt of the *parser grammar* for `g: Wikipedia`. The complete *parser grammar* has 74 lines (including comments etc.) and defines rules (e.g. `page`, `h1block`, `text`) for the syntactic analysis of a provided input. It is complemented by a *lexer grammar* of 109 lines, which specifies the rules (e.g. `TEXT`, `EQ`, `H1_OPEN`) for a preliminary lexical analysis—i.e. the tokenization of input. Please consider the lexer rule `h1`: it defines that the `title` of a level 1 heading is embraced by the lexer rule `H1_OPEN` (defined as `'\n='`) and `EQ` (`'='`)—the latter being also used in parser rules other than `h1` such as `text` in the presented excerpt.

[2]http://schereg.de.dariah.eu

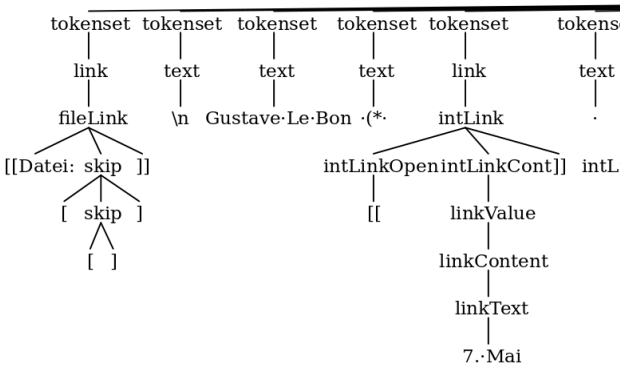

Figure 2: Parse tree of the wikipedia sample

Applying the grammars at the modeled element results in the generation of Java code and classes that represent a lexer, a parser, tree traversal helpers and auxiliary classes per defined rule. Upon requiring the grammar during the runtime of our developed system, these classes are dynamically loaded and hence—without any intervention by a programmer—the declarative definition of data in terms of DSLs results in the execution of native Java code that processes the defined data. As an end-result of the descriptive phase, a parse-tree is generated and reflects input data with respect to the specified grammar. Figure 2 shows the resulting parse tree of the first tokens of the textual content in listing 2.

Transformation functions

Comparable to its native context in compiler engineering, a parse tree reflects only an intermediate representation for the derivation of an enriched document and does not conclude our data enrichment process. Transformation functions form a subsequent step and are applied to parse trees. Listing 4 reflects the transformation function in `f: Wikipedia`. Opposed to the grammar, this function is not generic, but influenced by a specific application context in which only biographical sections are considered relevant. Whereas the first statement assigns any preface of an article to appropriate labels, the second statement contains a (simplified) filtering statement that is only satisfied, if the heading of a top-level (h2 in Wikipedia) heading contains the word 'life'.

Multiple transformation languages have been designed and standardized such as Query/View/Transformation (QVT), which is defined by the Object Management Group (OMG) [4]. Due to the complexity of such languages, we chose to define a simple transformation language that allows the specification of value and object assignments and command execution. To this point, the constraints of that language have not limited the use of the framework. However, as indicated in figure 3, the transformation language is merely

```
Section = @page.container.preface {
  Tokenset = @content.tokenset {
    Text = @[text, extLink.extLinkCont.linkValue];
    Wikilink = @intLink {
      LinkText = @linkValue;
      Link = @intLinkRes;
    };
  };
};
Section = @page.container.block.h2block[CORE::CONTAINS(
    @h2.title, "true", "life")] {
  Heading = @h2.title;
  Tokenset = @content.tokenset {
    Text = @text;
    Wikilink = @link {
      LinkText = @linkValue;
      Link = @intLinkRes;
    };
  };
};
AvailableHeading = @page.container.block.h2block.h2.title
    ;
```

Listing 4: Wikipedia function

another DSL and a set of user-selectable transformation languages could be integrated per request.

3.3 Rule processing overview

Figure 3 shows the overall concept of the transformation rule framework, which forms a combination of the application of DSLs for the description of data and a transformation language for the formulation of transformation rules. The building blocks within the framework are summarized in two interpreters, one for the validation and processing of input against a DSL and one for the validation and processing of a transformation function against the transformation language. Both are conceptualized as autonomous language application components—each accomplishing the tasks of processing input, constructing intermediate representations and traversing and rewriting these according to the needs of the concluding back end (*semantic analysis* or *optimization*). These *needs* are the named nodes and subtrees of parsed input and are known as soon as the transformation function has been parsed and analyzed.

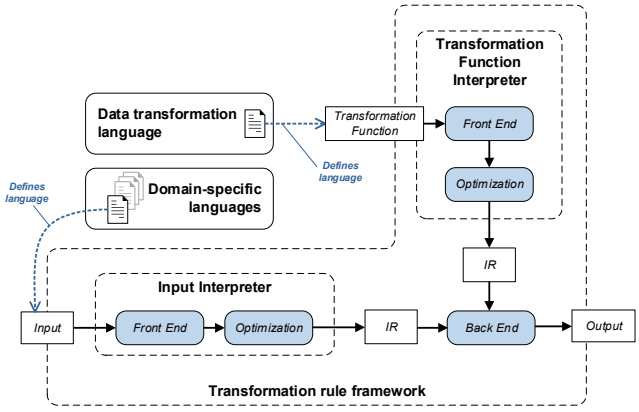

Figure 3: Transformation rule framework

At runtime of the rule framework, transformation functions are assumed to be reused for multiple instances of a defined element model. As such, only the content of individual documents needs to be interpreted, whereas the executable representation of the transformation function is cached in its intermediary state.

4. CONCLUSION

Although the example here touches all introduced topics, the results present only an intermediary step towards data that has been specifically modeled for a domain and research question. As subsequent steps, we have e.g. implemented functionality as usable commands for transformation functions to detect named entities by analyzing internal Wiki links or execute state-of-the-art methods of natural language processing (NLP)—to which the named entities that were identified by analyzing internal Wiki links can then be provided as known entities. With the help of the modeling framework we were able to compile an integrated data source for the prototypical implementation of a biographical analysis tool, which has been motivated by a feasibility study at the Leibniz Institute of European History in Mainz, Germany [2]. Despite the functional extensions to support NLP or Wiki link analysis functionality, no code was required to be written in order to access, process and transform data. These aspects have been solved by modeling data (along with the application context) in a declarative fashion, which not only resulted in a quicker prototypical implementation of the tool, but improved transparency and iterative adaptability for the domain experts.

5. REFERENCES

[1] T. Gradl and A. Henrich. A novel approach for a reusable federation of research data within the arts and humanities. In *Digital Humanities 2014 - Book of Abstracts*, pages 382–384, Lausanne, Switzerland, 2014.

[2] T. Gradl and A. Henrich. Nutzung und kombination von daten aus strukturierten und unstrukturierten quellen zur identifikation transnationaler lebensläufe. In E. Burr, editor, *DHd 2016*, pages 135–138. nisaba Verlag, 2016.

[3] M. Murata, D. Lee, M. Mani, and K. Kawaguchi. Taxonomy of xml schema languages using formal language theory. *ACM Transactions on Internet Technology*, 5(4):660–704, 2005.

[4] OMG. Meta object facility (mof) 2.0 query/view/transformation specification, 2015.

[5] T. Parr and K. Fisher. Ll(*): The foundation of the antlr parser generator. In *Proceedings of the 32Nd ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '11, pages 425–436, New York, NY, USA, 2011. ACM.

[6] M. Polfreman. Commonly-used metadata formats in the arts and humanities, 2005.

[7] P. Vierkant. Leuchttürme der deutschen repositorienlandschaft, 2013.

[8] Z. Zhang, P. Shi, H. Che, and J. Gu. An algebraic framework for schema matching. *Informatica*, 19(3):421–446, 2008.