# Leveraging Human Intelligence: Semi-automated Processing in Assuring Access to Digital Content

Mohammad Raza
Microsoft Research
21 Station Road
Cambridge CB1 2FB
+44 1223 479 700
a-moraza@microsoft.com

Natasa Milic-Frayling
Microsoft Research
21 Station Road
Cambridge CB1 2FB
+44 1223 479 700
natasamf@microsoft.com

## ABSTRACT
Need for standardization in the content production industry have led producers of popular authoring and publishing applications to adopt structured mark-up languages, such as XML, to implement their content file formats. As part of our effort to ensure long term access to such content, we need to consider properties of the mark-up schemas and devise methods to enable effective mapping among them. The methods may range from a fully automated mapping between two formats to semi-automated format transformation of individual artifacts through human intervention. The former is an ideal scenario and achievable when full specifications of the original and target formats are available and when the development of a full converter is economically feasible. However, a common lack of these resources creates challenges and requires exploration of alternative approaches.

To that effect, we propose a concerted research effort in formal characterization of the mark-up languages and the programming languages that can be used to express transformations of content and structures described through document mark-ups. Among these, we anticipate an important role for methods akin to programming-by-example where the document transformation is learnt by observing user interaction with the contemporary applications as the user manually performs changes in the document. Based on observed examples, the program then generalizes the transformations for the single document and similar documents in the corpora. Such approaches that leverage human input and effectively infer the desired transformations are essential for both creating and testing automatic document converters in general and handling a long tail of structured document formats for which the converters are not available.

## Categories and Subject Descriptors
H.3.0 [**Information Storage and Retrieval**]: General.

## General Terms
Algorithms, Reliability, Standardization, Languages, Verification.

## Keywords
Digital preservation, format transformation, XML, XML schema, mark-up language, programming languages, formal verification.

## 1. MOTIVATION
Many valuable digital artifacts are created using authoring and publishing tools that provide users with WYSIWYG interfaces as input and content presentation mechanism. Such tools are optimized for user productivity and have been widely adopted. However, the rapid evolution of these applications threatens our collective ability to provide the means of accessing and reusing content that has been authored by previous generations of such applications. One issue is that the legacy applications may not be compatible with the latest devices and the supporting software stack, e.g., the operating system, device drivers, etc. Thus, the original software cannot run in the contemporary computing environment. Another issue is the lack of user skills and information how to operate the legacy software. Both of these issues lead us to consider strategies that involve ingesting the legacy file content into the contemporary application that is familiar to the user and compatible with the contemporary computing environment. However, the content ingest is a complex matter in its own right: the application needs to have the ability to process the legacy file format and extract relevant information for its own processing.

Standardization efforts in the content production industry have led a number of popular authoring and publishing applications to adopt XML mark-up language as the basis for implementing their content file format. Thus, our considerations of file ingest and format transformations need to consider XML or similar mark-up schemas and the methods to enable effective mapping between such formats. Indeed, the ingest of a document from one application into another that uses a different document schema is likely to involve automated mapping between the two mark-up schemas when it is feasible to implement the full file format converter. However, in practice, the format specifications may not be known for the formats involved or the cost of producing a full converter may not be justified due to the low volume of documents to be processed.

For these reasons, we propose a concerted effort to develop methods that leverage the human intelligence and skills in using contemporary applications and the power of computing to learn desired transformations by observing user interactions and then automating them.

In our research we have investigated semi-automated methods for XML format transformation and based on that experience we propose to investigate several specific aspects of structured formats that use mark-up languages like XML. In the following sections we reflect on the state-of-the-art in methods for learning from user interaction and discus the outstanding issues

## 2. CURRENT STATE OF THE ART

### 2.1 Contemporary Authoring Applications
Applications for authoring digital documents such as Microsoft Word and Open Office Word processing software provide editing and formatting functions that the user can apply to content as displayed on the screen. The results of the user actions are

immediately presented on the screen and the changes are persisted in the document file as the user saves the document. Files produced by contemporary office productivity tools use XML mark-up language to store information about the content and structure of the document. Such are two adopted XML standards for word processing files, Open Document Format (.odt) and OpenXML format (.docx).

While some of the editing features apply to the global document formatting, such as page margins, multi-column specifications, and line spacing, many functions are focused on localized transformations of content where the user first selects the content range and then applies a particular function. Editing can thus be viewed as a sequence of local transformations of XML files using program functions that are instantiated from the menus or by short-cut key strokes. While the applied functions are not recorded, the results of the transformations are reflected in the XML encoding that describes the document content and format. In that sense, the XML mark-up represents the document state arrived at through transformations.

## 2.2 Learning from Human Interactions

We assume a scenario in which the user is presented with a file in an XML based format with unknown document schema and uses MS Word application to open it. MS Word can load and recognize the XML structure but cannot interpret the formatting specifications. Thus, the user has to start manually editing the document using the MS Word features. From the user's actions, MS Word would automatically infer the mapping of the unknown schema onto the MS Word .docx format and propagate the changes wherever they apply in the document. Furthermore, the transformations inferred from this document could be applied to similar document in the corpus.

Previous research explored learning format transformations for XML documents using machine learning techniques such as tree transducers [1], probabilistic context free grammars [2] and conditional random fields [3]. However, they are generally based on the availability of large sets of training data in the form of collections of documents in both the source and the target format. These techniques are hence not applicable in the end-user setting where such data may not be available.

A more suitable paradigm for the described scenario would be programming-by-example (PBE) in which a general program, e.g., a converter between XML formats, can be learnt from specific examples demonstrated by a user. This area has been gaining renewed interest [4] because of the potential to enhance the productivity of end-users in a variety of common tasks performed using authoring tools.

Recent work in the PBE includes syntactic string transformations in spreadsheets [6], semantic string transformations [7], and techniques for automating repetitive data manipulation tasks [5]. We have considered a generalization of such work to deal with manipulating complex XML formats that can be represented as trees with elaborate structures, such as tables. In terms of algorithmic technique we use approaches based on the version space algebra for the replace-maps as in [6]. However, we consider richer loop structures that involve iterators to capture a broad range of edits and enable efficient synthesis of edits using ideas of angelic programs.

A closely related area is Programming by Demonstration (PBD) in where, instead of input-output examples, the user provides a demonstration trace of actions. The demonstration is, in essence, a sequence of editor states after each primitive action, showing how to do the transformation to achieve a particular output. Examples of PBD for text editing include SmartEdit [8], TELS [10] and Lapis [9].

In PBD systems the user is typically required to segment the loops and indicate each iteration. Furthermore, the programs generated by PBD have the drawback of being dependent on the order in which the user chooses to perform actions. We prefer the PBE over the PBD approach as PBE requires the user to provide only the final state, i.e., the desired output, instead of providing the intermediate states. That increases the utility of our system but at the expense of making the synthesis problem much more difficult. Thus, the focus of the investigation becomes to find efficient procedures to deal with that issue.

# 3. RESEARCH CONTRIBTIONS AND BENEFITS

We have conducted research with the .docx XML format and gained valuable insights into the technical aspects that need to be addressed. In particular, we have developed a method that enables the user to perform the desired content transformations on a couple of examples and automate content edits across other instances in the document. For example, the user may want to reformat a bulleted list of 35 names and addresses into a table comprising several columns of data. The user can show the desired transformation on two or three examples and the program will learn and apply it to the consecutive instances in the list.

We specified the problem as a program inference task: from the XML structure and mark-up of a given document and from the edited XML example, we infer the program that transforms the original XML to the edited XML and correctly applies the same transformation to the rest of the document.

In order to define the space of plausible programs, we define a domain specific language for XML transformations and design an algorithm to infer transformations in this language, given a source and target example of a format transformation. The language of transformations is defined on *factor trees* rather than on XML trees directly. A factor tree is an intermediate tree that represents the classification of content in the original XML tree. We consider the problem of program inference on factor trees under the premise that format transformations preserve the content in a document.

The domain specific language (DSL) of transformations is based on the search-replace paradigm, in which the source tree can be partitioned into disjoint sub-graphs based on a search filter, and these sub-graphs can be transformed independently of one another using a replace operation. The algorithm infers such search-replace operations from the given source and target examples. Search filters are determined by matching sub-graphs of the source and target trees based on the classification of content nodes in the two trees and matching nodes that classify the same content. For each pair of sub-graphs matched, the algorithm synthesizes a general replace transformation using a method based on version space algebras similar to the one presented in [6].

In order to cover a broad range of formats, content types, and application scenarios, we propose a research agenda to include:

- Investigation of the mark-up languages used for structured document formats in order to identify commonalities and differences.
- Development and evaluation of the domain specific languages (DSLs) to support the representation and the inference of format transformations.
- Explorations of trade-offs between the expressivity of the DSLs and the computational costs of inference as the scope of possible transformation expands with increased expressivity.
- Development and evaluation of the user interaction features to support the semi-automatic editing of structured documents.

Besides scenarios of semi-automated format transformations, the same type of methods could be used to support implementation and testing of the automatic document transformers. Whether such transformers are rule based or learned by applying machine learning techniques to large collections of test documents, the resulting transformers are unlikely to be 100% accurate. Thus, adding PBE techniques to collect information about the desired transformations directly from the users would be beneficial in cases when there are insufficient examples in the test corpora.

## 4. OUTLOOK

We believe that a concerted research effort in formal characterization of mark-up languages and a range of DSLs designed to express transformations of document formats is fundamental to our ability to access legacy content through contemporary applications. At the same time, the principles of PBE enable us to leverage the human input and combine it effectively with the computing automation. The outcomes of this research would be critical for both testing and improving the document converters that are in high demand and for evolving the conversion capabilities for the long tail of formats without existing convertors.

## 5. REFERENCES


[1] S. Maneth, A. Berlea, T. Perst, H. Seidl. XML type checking with macro tree transducers. In: *24th ACM Symposium on Principles of Database Systems*, 2005, p. 283–294.

[2] B. Chidlovskii, J. Fuselier. A probabilistic learning method for XML annotation of documents. *In: Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, 2005, p. 1016-1021.

[3] Rémi Gilleron, Florent Jousse, Isabelle Tellier, Marc Tommasi: XML Document Transformation with Conditional Random Fields. *In: INEX 2006*: 525-539

[4] Sumit Gulwani: Synthesis from Examples: Interaction Models and Algorithms. *In: SYNASC 2012*: 8-14

[5] William R. Harris, Sumit Gulwani: Spreadsheet table transformations from examples. *In: PLDI 2011*: 317-328

[6] Sumit Gulwani: Automating string processing in spreadsheets using input-output examples. *In: POPL 2011*: 317-330

[7] Rishabh Singh, Sumit Gulwani: Learning Semantic String Transformations from Examples. *In: PVLDB* 5(8): 740-751 (2012)

[8] T. Lau, S. Wolfman, P. Domingos, and D. Weld. Programming by demonstration using version space algebra. *In: Machine Learning*, 53(1-2), 2003.

[9] R. C. Miller and B. A. Myers. Interactive simultaneous editing of multiple text regions. *In: USENIX Annual Technical Conference*, 2001.

[10] I. H. Witten and D. Mo. TELS: learning text editing tasks from examples. *In: Watch what I do: programming by demonstration*, pages 293–307. MIT Press, Cambridge, MA, USA, 1993.