

An Efficient Tool for XML Data Preparation

Kazem Taghva, Marc Cartright, and Julie Borsack
Information Science Research Institute
University of Nevada, Las Vegas

1. INTRODUCTION

Processing massive amounts of unstructured textual information is one of the most critical requisites of today's computers. Unfortunately, selecting, processing, and interpreting textual data is not as explicit as the number crunching tasks computer's were originally designed to solve. Most, if not all text analysis systems require some amount of data for training before they are useful for a particular application. Preparing good training data, although tedious and expensive, is highly correlated to the quality of results. So simplifying and perfecting the data preparation task is imperative for all data analysis systems.

The *Meta-marker* program described in this paper serves to quickly prepare training data from unstructured text for text analysis applications. The system applies eXtensible Markup Language (XML) in a way that results in consistent training data while still providing the flexibility expected for such a diverse class of applications.

2. BACKGROUND AND RELEVANT RESEARCH

Unstructured text analysis includes a wide range of applications including document mining (DM), information extraction (IE), natural language processing (NLP), knowledge acquisition (KA), and document summarization (DS). Further, for each of these classes of applications, there are an infinite number of domains and data specifications that dictate the attributes of the training material. For example, an information extraction task may require the identification of scenarios that disclose sensitive homeland security information. A very different IE task would be to identify personal home addresses. Each would have very different training data requirements.

As noted in the Introduction, an expensive yet necessary aspect in the development of text analysis systems is the preparation of training data. Sattler and Schallehn¹ state that “. . . 50-70 percent of the time and effort in data analysis projects is required for data preparation” This estimate would not be argued by anyone who has performed text analysis research.

Most unstructured text analysis projects in the literature make one of two assumptions: 1) the text to be analyzed is *semi-structured* and already has some markup, i.e., HTML in web pages^{2,3} or 2) a small amount of training data will suffice or will be used to generate more training data automatically.⁴ These are reasonable assumptions for certain tasks but in general, specific training examples are called for. There has been at least preliminary work performed in the area of data preparation for data mining.^{1,5,6} In all these works, the process of data preparation is broken down into subtasks:

- Data Selection
- Data Integration (Construction)

- Data Transformation (Formatting)
- Data Cleaning
- Data Reduction

Several of these subtasks transfer directly to the preparation of training data from unstructured text as well. The Meta-marker program focuses on *data selection*, *data integration* and *data transformation*. We illustrate how the program addresses these subtasks in Section 3. In general, the Meta-marker program addresses the issue of preparing training data from unstructured text. Since the development of training material cannot be fully automated, a semi-automated approach is the most we can hope for to increase the efficiency and accuracy of data preparation.

3. DESIGN AND ARCHITECTURE

The Meta-marker concept was formulated by our environment as much as our need for data. Our research environment consists of large collections of unstructured full text documents and their corresponding page images. An image page with its generated text provides a functional user interface (the image) as well as the ASCII for processing and analysis. What's important is to coordinate these two mediums to work as a single unit. Much of this work was pioneered at the Information Science Research Institute (ISRI) in the early 1990's. Section 3.1 gives a brief description of prior systems that established the coordinated image/text processing applied in Meta-marker. The subsequent Subsections describe other architecture and implementation decisions made in the design of the system.

3.1. ISRI's Prior Work

Two systems in particular advanced the image/text coordination applied in Meta-marker. The first is *Autotag*,⁷ a system that was designed to automatically convert printed collections into their tagged electronic versions. Autotag accepts a physical document representation, i.e., pages, zones, strings, and text features from an OCR device, and outputs a logical document representation made up of words, sentences, paragraphs, sections, title, etc. So the notion of using special text features provided by an OCR device to tag logical information originated with Autotag.

MANICURE is another OCR post-processing system developed at ISRI⁸ that had a direct influence on the design of the Meta-marker system. First, MANICURE generates the XML input format (1doc) that provides a word's x and y coordinates on an image. This information gives Meta-marker the ability to highlight bounding boxes on the image and tag the corresponding words in the text. Further, MANICURE applies a similar user interface to allow semi-automated OCR correction. MANICURE automatically corrects as many misrecognized words as possible but it also provides a semi-automated tool, the MANICURE Correction Tool (MCT), that displays an image page and highlights misspellings so that a user has the ability to bring a document to an even higher level of accuracy if required.

3.2. Implementation Decisions

The Meta-marker program is written primarily in the scripting language Ruby.⁹ The impetus for this decision was that 1) Ruby works extremely well as an object-oriented language, thereby providing an easy transition from model to implementation, 2) Ruby allows for rapid development and debugging of the software, and 3) should we decide to port the program to another platform, such as Windows, the effort would require significantly less time than using a compiled language. To provide a windowing toolkit for the program, we use a modified version of the FXRuby¹⁰ library, originally designed by Lyle Johnson.

The advent of XML provided a standardized method for explicitly tagging data in unstructured text. For this reason, the Meta-marker exclusively uses XML for its output formats. However, formatting the data in XML manually is impractical for anything longer than a few lines. This is the fundamental purpose of the Meta-marker: to automate the process of structuring data in XML. The system's purpose is to minimize the preparer's role in the most repetitive and error-prone aspects of data preparation. Instead of manually typing the XML, the preparer can use a GUI and a mouse to mark the data, making it possible to prepare a larger amount of data in a smaller amount of time. By automating the markup process, we also reduce the chance of error from fatigue and repetition that would be encountered in manual preparation.

3.3. Design

The design of the Meta-marker aims to separate the generalities in the theory of data preparation from the specifics encountered when implementing a preparation system. To effectively implement the *data selection* aspect of the data preparation task, only a few basic elements were required:

1. A way to efficiently represent the data to the preparer.
2. A way to denote the semantics and syntax of the metadata to apply to the data.
3. A way to efficiently allow the preparer to apply the metadata to the representation.

If we consider that the term “data” applies to all relevant data for the system to be trained, regardless of the source of that data, then these elements also effectively implement the idea of *data integration*. The Meta-marker is flexible enough to integrate training data from disparate sources and by modeling the output process similarly to the input process, we gain the ability to output the prepared data in a variety of formats. The insertion of XML tags facilitates the *data transformation* part of the data preparation task.

3.4. Architecture

The Meta-marker program is hierarchical in structure. At the most general level, we have the Meta-marker *Shell* itself which plays host to the various types of media classes that present the data to the end user. The Meta-marker is only aware of a few concepts: a medium, the selector instances, and a document contained within the medium. The shell requires very little modification, if any, when modifying the program, as is intended.

The *Media* classes reside just below the shell in the hierarchy. Here we decide what serves as an appropriate working representation of the data and the behavior of the program with respect to that data. The ImageCanvas class acts as a medium to display image

files (currently only TIFF files are supported) using some intermediary format to supply the coordinates of the interesting constructs of the image loaded. Another medium is the TextCanvas class which displays ASCII text documents.

Working in concert with the media classes are the *Page* classes. While a specific medium defines the behavior of the program, a specific page defines exactly what data the medium will have available. For example, the page implementations for the TextCanvas medium only need to provide a text string for the medium and be able to track selections made on the medium. However, the ImageCanvas medium must manipulate an image as well as track selections.

The *Codec* classes serve as interfaces between the file system and the pages displayed in the Meta-marker interface. Generally, a single codec handles the loading and saving of a particular format. However, in some instances the codec only provides a one-way operation. For example, one codec could be used to load emails stored in a particular format but saving into that format may not make sense since most email formats do not provide a facility for storing metadata that is not useful to a mailing agent. The codecs are the most specific components of the program.

The *Selector* class does not really exist in the hierarchy. The selector class will never need to be subclassed because it merely acts as a representation of meta-information a preparer would want to apply. An oversimplified analogy to multiple selector instances would be a set of differently colored highlighters - you may have yellow represent an important concept in a document, while green indicates a useful reference, and blue indicates any claim that you may find disputable or incorrect. Each color, when applied to some part of the document, indicates something of interest to the user. The idea of the selector is also independent of the input and output format. The end user will want to identify interesting pieces of data, and the selectors serve this purpose.

Any extensions to the program will adhere to this architecture since it provides the framework for effectively applying meta-information to data.

4. ILLUSTRATIVE TAGGING APPLICATIONS

As mentioned in Section 3, The bulk of our research at ISRI is performed on unstructured full text documents. These documents are imaged, OCR'd, and automatically post-processed providing textual files with corresponding image pages. In recent years, our work has been focused on identification and extraction of relevant information from these documents. In particular, since the documents were intended for public consumption, identification of sensitive scenarios (Homeland Security) and violations of individual privacy (Privacy Act of 1974,¹¹ Freedom of Information Act¹²) were expected prior to any document's release. This kind of "document mining" called for a quick and easy way to build the various types of training data required for eventual automation of these tasks. Sections 4.1 and 4.2 review two text preparation tasks to which the Meta-marker was directly applicable.

4.1. Ground Truth for an HMM

Since most of the unstructured text in our research is generated from hardcopy documents, the original page images are available for document processing. This is important for two reasons: 1) the image can be used as part of the data capture process and 2) the generated text may potentially contain OCR errors making address identification more difficult.

Having the image pages available influenced the design of the Meta-marker system. It necessitated the notion of a media class and provided a much more practical interface for users.

One piece of information protected under the Privacy Act is an individual's personal home address. Our solution was to apply a Hidden Markov Model (HMM)¹³ for identifying address patterns.¹⁴ An HMM requires training examples to learn the various address patterns and to estimate state transition probabilities. The more examples provided, the more accurate the HMM.

Figure 1* shows the Meta-marker interface and the way a user would "tag" relevant pieces of information on an image page. The selectors for address data collection appear in the second row of the window interface. Note that the selectors are fully customizable. For this particular application, only five selectors were required but any number and any type of selectors can be defined. A selector is chosen and then the bounding boxes for the words are "mouse clicked," indicating this selector is bound to the selected word. A word can be assigned multiple selectors. Below is a portion of the generated XML where the address has been tagged for training.

```
<hmm_segment>
<markdoc>/metamarker/hmm-home-address/sample.ldoc</markdoc>
. . .
<background>Project</background>
<background>Office</background>
<background>Richland</background>
<background>Operations</background>
<background>Office</background>
<prefix1>U.S.</prefix1>
<prefix2>Department</prefix2>
<prefix3>of</prefix3>
<prefix4>Energy</prefix4>
<target1>P.O.</target1>
<target2>Box</target2>
<target3>550</target3>
<target4>Richland,</target4>
<target5>WA</target5>
<target6>99352</target6>
<suffix1>Thank</suffix1>
<suffix2>you</suffix2>
<suffix3>for</suffix3>
<suffix4>your</suffix4>
<background>interest.</background>
<background>Please</background>
<background>be</background>
<background>assured</background>
<background>that</background>
. . .
</hmm_segment>
```

Hundreds of address examples were prepared in just a few hours using the Meta-marker program and by using the simple GUI interface, human error was minimized.

*A government address, which is *not* protected under the privacy act was used in this example.

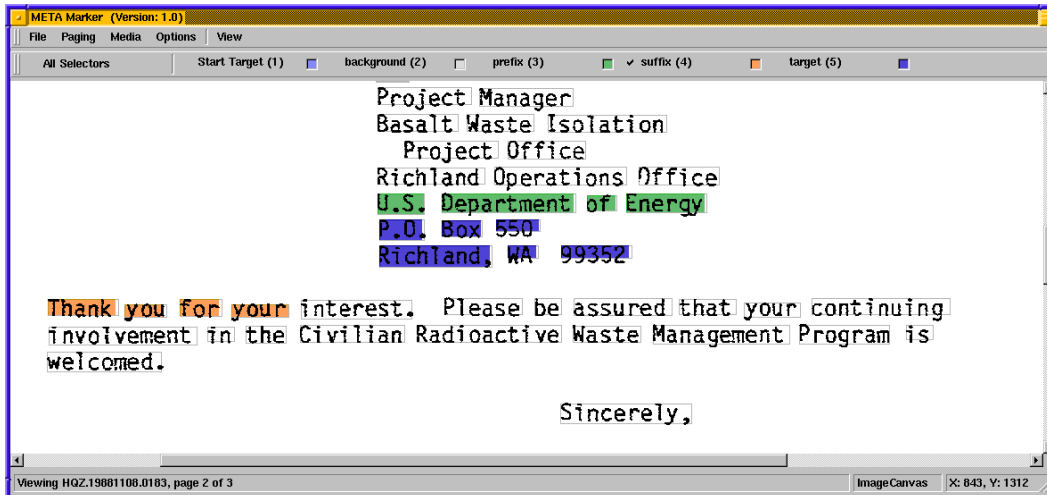


Figure 1. GUI Interface for ImageCanvas Markup

4.2. Lexicon of Privacy Data

The Privacy Act of 1974 protects not only home addresses but also a myriad of other personal information including uniquely identifiable information (e.g. social security number, passport number), education and employment history, financial information, medical information, family facts, personal preferences and affiliations. Another method we applied to discover personal privacy in unstructured text was through the identification of “features.” For example, certain words or combinations of words, tend to indicate disclosure of privacy. Building a lexicon of privacy words and patterns that occurred in our corpus was the initial phase of designing our automated system, the Privacy Act Classifier (PAC).¹⁵ Once the selectors are determined for a particular application, the Meta-marker can be efficiently applied to collect and tag the data. Figure 2[†] displays a text email with some tagged privacy information. With the Meta-marker GUI, highlighting text files is equivalent to highlighting images. Further, the output for both media types is identical, simplifying data processing. Below is the generated XML with all the appropriate privacy markup.

```
<hmm_segment>
<markdoc>/homebase/isri/jborsack/cvs/isri/papers/metamarker/privacy-example.xml</markdoc>
<place_of_birth>Place</place_of_birth>
<place_of_birth>of</place_of_birth>
<place_of_birth>Birth</place_of_birth>
<place_of_birth>(City,</place_of_birth>
<place_of_birth>Country):</place_of_birth>
<place_of_birth>Annemasse,</place_of_birth>
<place_of_birth>France</place_of_birth>
<date_of_birth>Date</date_of_birth>
<date_of_birth>of</date_of_birth>
<date_of_birth>Birth:</date_of_birth>
<date_of_birth>01/23/1971</date_of_birth>
```

[†]Privacy information was modified for this example.

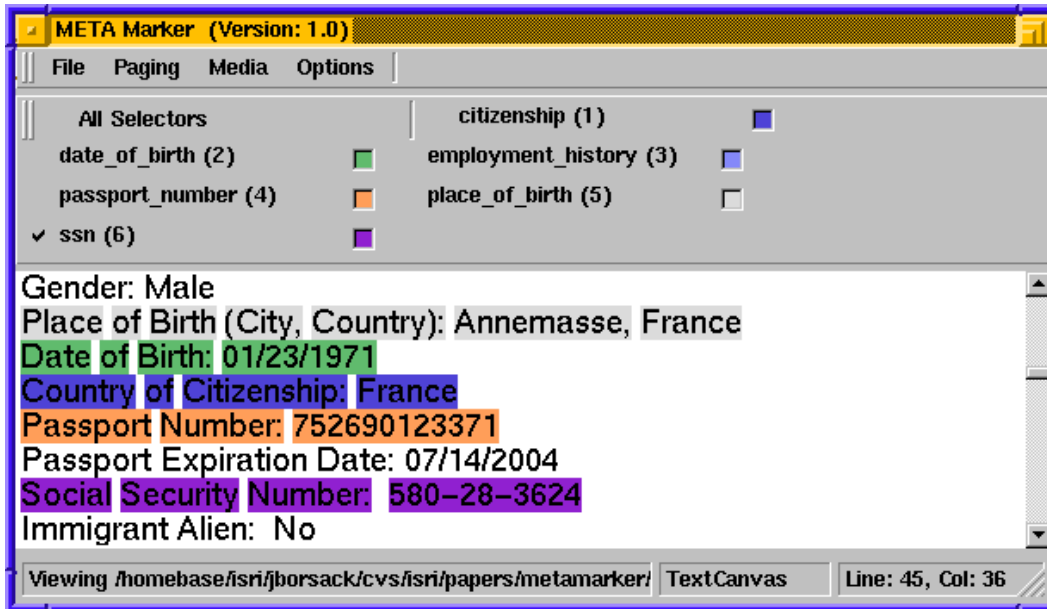


Figure 2. GUI Interface for TextCanvas Markup

```

<citizenship>Country</citizenship>
<citizenship>of</citizenship>
<citizenship>Citizenship:</citizenship>
<citizenship>France</citizenship>
<passport_number>Passport</passport_number>
<passport_number>Number:</passport_number>
<passport_number>752690123371</passport_number>
<ssn>Social</ssn>
<ssn>Security</ssn>
<ssn>Number:</ssn>
<ssn>580-28-3624</ssn>
</hmm_segment>

```

5. FUTURE DIRECTION

Based on the applications to which we have applied the Meta-marker, the system already shows great potential for reducing the amount of time and effort required for unstructured data preparation tasks. There are though several enhancements that we believe would make the Meta-marker more general.

First, to make the application completely customizable by a non-technical user, the creation of an XML-based selector file should be simpler. Currently, for a user to create a new selector file, he would need at least an example selector file and a basic understanding of XML. A separate module is under development to provide a graphical interface to prepare the selector files. The module will operate separately from the main program as well operate as a sub-window of the main program.

A second consideration that may hinder the Meta-marker's wide-spread applicability is a lack of context between the selector instances themselves. Currently, it is possible to tag a data item with multiple selectors, thereby marking one piece of information with multiple tags. The program though has no ability to allow one selector to take precedence over another. In some applications relationships among selectors may be essential. There is no mechanism currently under development to address this problem, but the notion of a hierarchical relationship among selectors is promising. By using these concepts it is possible to create a partial order between the selectors where the leaf nodes would be applied to the data first, continuing up the order, and applying the root node last.

Another condition that we plan to consider is the notion of *scope*. In Meta-marker v1.0, there is no mechanism in place to define *how* data should be tagged outside of the context of media. For example, if one were to define a set of selectors **Alpha**, **Digit**, and **Space** to tag each character and whitespace in a document, we run into a conundrum: which of the following representations is correct?

<code><Alpha>The</Alpha></code>	<code><Alpha>T</Alpha></code>
<code><Space> </Space></code>	<code><Alpha>h</Alpha></code>
<code><Alpha>horse</Alpha></code>	<code><Alpha>e</Alpha></code>
.	or <code><Space> </Space></code>
.	<code><Alpha>h</Alpha></code>
.	<code><Alpha>o</Alpha></code>
	<code><Alpha>r</Alpha></code>
	<code><Alpha>s</Alpha></code>
	<code><Alpha>e</Alpha></code>

In order to convey the markup on the right, each character would need to be tagged separately. This method is obviously tedious and inefficient. By introducing the notion of scope, this ambiguity will be avoided. Currently, scope has three valid values: **character**, **word**, and **phrase**. So for selector sets where heterogeneous levels of semantics exist, knowing the scope of each selector is essential to proper tagging of the data.

We also plan to add support for different input and output formats as the need arises. While the Meta-marker has proven itself in the most common training data requirements, by addressing the issues mentioned above, we hope to integrate the program as the primary method of data preparation for the projects at ISRI.

REFERENCES

1. K. Sattler and E. Schallehn, "A Data Preparation Framework based on a Multidatabase Language," in *Proc. of Int. Database Engineering and Applications Symposium (IDEAS 2001)*, M. Adiba, C. Collet, and B. P. Desai, eds., pp. 219–228, IEEE Computer Society, (Grenoble, France), 2001.
2. O. Etzioni, "The world-wide web: Quagmire or gold mine?," *Communications of the ACM* **39**(11), pp. 65–68, 1996.
3. S. Soderland, "Learning to extract text-based information from the world wide web," in *Knowledge Discovery and Data Mining*, pp. 251–254, 1997.
4. S. B. Huffman, "Learning information extraction patterns from examples," in *Learning for Natural Language Processing*, pp. 246–260, 1995.

5. J. Han and M. Kamblar, *Data Mining: Concepts and Techniques*, ch. 3. Morgan Kaufman Publishers, 2001.
6. CRISP-DM Consortium, *CRISP-DM 1.0: Step-by-step data mining guide*, 2000. <http://www.crisp-dm.org>.
7. K. Taghva, A. Condit, and J. Borsack, "Autotag: A tool for creating structured document collections from printed materials," in *Electronic Publishing, Artistic Imaging, and Digital Typography, Proc. of the EP '98 and RIDT '98 Conferences*, R. D. Hersch, J. Andre, and H. Brown, eds., *Lecture Notes in Computer Science*, pp. 420–431, Springer-Verlag, (St Malo, France), April 1998.
8. K. Taghva, A. Condit, J. Borsack, J. Kilburg, C. Wu, and J. Gilbreth, "The MANICURE document processing system," in *Proc. IS&T/SPIE 1998 Intl. Symp. on Electronic Imaging Science and Technology*, (San Jose, CA), January 1998.
9. webmaster@ruby-lang.org, "The Ruby home page." <http://www.ruby-lang.org>.
10. lyle@users.sourceforge.net, "The FXRuby." <http://fxruby.sourceforge.net>.
11. "The Privacy Act of 1974, as amended." 5 U.S.C. §552a, 1974. <http://www.usdoj.gov/04foia/privstat.htm>.
12. "The Freedom of Information Act, as amended in 2002." 5 U.S.C. §552, 2002. <http://www.usdoj.gov/04foia/foiastat.htm>.
13. D. Freitag and A. McCallum, "Information extraction with HMM structures learned by stochastic optimization," in *AAAI/IAAI*, pp. 584–589, 2000.
14. K. Taghva, J. Coombs, and R. Pereda, "Address extraction using hidden markov models." For submission to Proc. IS&T/SPIE 2004 Intl. Symp. on Electronic Imaging Science and Technology, July 2004.
15. Information Science Research Institute, *Privacy Act Classifier (PAC) User Guide*, version 1.2 ed., 2004.