Assisted Structured Authoring using Conditional Random Fields

Bert Willems
FontoXML
<bert.willems@fontoxml.com>

Abstract

Authoring structured content with rich semantic markup is repetitive, time consuming and error-prone. Many Subject Matter Experts (SMEs) struggle with the task of applying the correct markup. This paper proposes a mechanism to partially automate this using Conditional Random Fields (CRF), a machine learning algorithm. It also proposes an architecture on how to continuously improve the CRF model in production using a feedback loop.

Keywords: XML, Conditional Random Fields, Structured Authoring, Machine Learning

1. Introduction

With the increasing adoption of structured XML content, the amount of work required from Subject Matter Experts (SMEs) increases. Not only are they required to capture their knowledge as information to others, they are increasingly asked, and sometimes even required, to mark up the information with the appropriate semantic and structural metadata in the form of XML tags and attributes.

Examples of those markup tasks include:

- Structuring bibliographic references to tag authors, journal name, publisher etc.
- Marking up tasks, not with ordered lists but with steps.
- Marking up interactive questions, like multiple choice questions.

Although WYSIWYG XML editors help to make this task as easy as possible, the fact remains that there is additional work to be done that is often repetitive and error-prone. FontoXML conducted multiple studies to determine whether the effort of manual tagging affected adoption. The results showed a consistent negative effect: SMEs and their editorial colleagues are hesitant to adopting structured authoring. In some cases this meant reverting back to their unstructured content processes, leading to unrealized potential.

Prior implementations, like GROBID [3], apply markup automatically. This paper proposes to introduce Machine Learning (ML) to the authoring process

instead. The reason for this is the inaccuracy of the state-of-the-art ML algorithms: like humans, they make mistakes. Allowing SMEs to (correct and) accept a machine provided suggestion will result in a more accurate markup. Furthermore, this approach allows for the creation of a feedback loop, allowing the machine to improve over time.

This paper focuses on the task of structuring bibliographic citations, although the proposed architecture scales to many of the tasks required for properly structured content.

2. Model

This section describes the model used for recognizing bibliographic citations and extracting the relevant labels from it. The model used in this paper follows a divide-and-conquer strategy and is made up out of two separate models: The Citation Model and the Name Model. Partial results from the Citation Model cascade into the Name model to more detailed results.

2.1. Citation Model

The goal of the Citation Model is to classify a sequence of text with tags that make up the parts of the citation. The tags are derived from the TEI P5 vocabulary [12] and are encoded using the IOB tagging scheme [8].

The following tags are distinguished:

- author
- orgName
- editor
- publisher
- pubPlace
- date
- idno (bibliographic identifier)
- analytic (articles, poems, etc.)
- monographic (books, single & multi volumes, etc.)
- journal
- series
- unpublished
- volume
- issue
- pages
- chapter

For example, the sequence *Erickson, T. & Kellogg, W. A. "Social Translucence: An Approach to Designing Systems that Mesh with Social Processes." In Transactions on*

Computer-Human Interaction. Vol. 7, No. 1, pp 59-83. New York: ACM Press, 2000. is tagged as[1]:

author	Erickson, T. & Kellogg, W. A.
analytic	Social Translucence: An Approach to Designing Systems that Mesh with Social Processes
journal	Transactions on Computer-Human Interaction
volume	7
pages	59-83
pubPlace	New York
publisher	ACM Press
date	2000

2.2. Name Model

The Name Model is much simpler[2] compared to the Citation Model. The purpose of the Name Model is to distinguish names in a given sequence of text. To be more specific in the *author* and *editor* labels cascading from the Citation Model. The tags are also encoded using the IOB tagging scheme.

The following tags are distinguished:

- forename
- middlename
- surname

For example, the sequence *Erickson, T. & Kellogg, W. A.* is tagged as:

surname	Erickson
forename	T
surname	Kellogg
forename	W.
middlename	A.

[1]The IOB prefixes are combined and the outside tags are removed in the example output for brevity.
[2]Although the model is comparatively simple, handling names is surprisingly complex. See: http://www.kalzumeus.com/2010/06/17/falsehoods-programmers-believe-about-names/ . The point of the model is to learn from examples in the training data rather than manually coded rules.

2.3. Linear Chain Conditional Random Field

Both the Citation and the Name model are implemented using the Conditional Random Fields (CRF) algorithm [7]. According to the research performed by Fuchun Peng and Andrew McCallum [10], CRFs work well for extracting structured bibliographic citations from research papers, achieving good performance. This algorithm is also used in similar implementations like GROBID and MALLET [9].

To get a sense of how CRFs work, consider the sentence "I'm at home.". Now consider the sentence "I'm at kwaak.". Based on both sentences one intuitively understands that "kwaak" is some sort of location because we know that "home" is also a location and the words appear in the same *context*.

CRFs take into account the context in which a word appears and some other features like "is the text made up out of numbers?". More precisely: an input sequence of observed variables X represents a sequence of observations (the words with the associated features which make up a sentence) and Y represents a hidden (or unknown) state variable that needs to be inferred given the observations (the labels). The Y_i are structured to form a chain, with an edge between each $Y_{(i-1)}$ and Y_i. As well as having a simple interpretation of the Y_i as "labels" for each element in the input sequence, this layout admits efficient algorithms for:

1. model training, learning the conditional distributions between the Y_i and feature functions from some corpus of training data.
2. decoding, determining the probability of a given label sequence Y given X.
3. inference, determining the most likely label sequence Y given X.

For a more detailed introduction to CRFs, see An Introduction to Conditional Random Fields for Relational Learning [11].

2.4. Implementation

For the implementation of the CRFs, an implementation based on CRFSharp [1] is used. CRFSharp is a .NET Framework 4.0 implementation of Conditional Random Fields written in C#. Its main algorithm is similar with CRF++ written by Taku Kudo [6]. It encodes model parameters by L-BFGS. Moreover, it has many significant improvements over CRF++, such as totally parallel encoding and optimized memory usage. The CRFSharp implementation was modified to target the .NET Standard 2.0 to allow cross-platform usage in .NET Core applications.

2.5. Training

In order for the CRFs to learn the desired outcomes, they must be trained first using correctly tagged example inputs. For the training of the Citation Model, a corpus of 439 TEI documents is used originating from the GROBID GIT reposi-

tory. The corpus contains 10.288 properly marked up bibliographic citations (<bibl/>). 6.467 citations are used for training, while the remaining 3.821 are used for evaluation. For the training of the Name Model, a corpus of 13 TEI documents is used originating from the GROBID GIT repository. The corpus contains 403 properly marked up names of which 370 names are used for training, and the remaining 33 names are used for evaluation.

Both models were trained with a maximum iteration count of 1000 using L2 regularization running on 8 threads in parallel. The training of the Name Model takes no more than a minute while training the Citation Model takes around 2,5 hours on a laptop with an Intel i7-4702HQ processor and 16GB of RAM. Training was clearly CPU bound; the 8 logical processors were 100% utilized. Memory usage was around 1.5 GB. Training speed can possibly be improved using the GPU rather than the CPU. However this was not explored.

2.6. Model Evaluation

The trained models are evaluated against previously unseen examples. Both models are scored on the overall performance of all their labels. Both models are scored using the accuracy and F_1 metrics which are common scoring metrics for Named Entity Recognition (NER) models. The scoring is defined as:

1. Accuracy; Where TP is the number of true positives, FP is the number of false positives, TN is the number of true negatives and FN the number of false negatives. Accuracy is calculated as: $accuracy = \frac{TP + TN}{TP + FP + TN + FP}$.
2. F1 measure; Precision, recall and F1 measure are defined as follows: $precision = \frac{TP}{TP + FP}$, $recall = \frac{TP}{TP + FN}$, $F_1 = 2 \times \frac{precision \times recall}{precision + recall}$.

Both models perform reasonably well on the evaluation dataset. Table 1 shows the results of the evaluation of both models:

Table 1. Evaluation Results

Model name	**Accuracy**	**F_1**
Citation Model	99,53%	99,72%
Name Model	99,36%	99,31%

One of the observations made during the evaluation is that the Citation Model has a hard time predicting the correct type for the titles (analytic vs monographic and journal vs series vs unpublished). Similarly, it has a hard time distinguishing between author and editor. This is partly due to the local nature of CRFs; the CRFs in this application use a context window of 4 words around the target word. Introducing a bidirectional Long Short-Term Memory (LSTM) in combina-

tion with a CRF layer overcomes this local nature [13]. Another boost may be a gained from introducing lexicons for journal titles etc.

2.7. XML Mapping

Mapping the extracted labels to an XML vocabulary is straightforward, see Table 2 for a mapping between the labels and the JATS [5] & TEI vocabularies:

Table 2. Labels to XML mapping

Label	**JATS `<mixed-citation/>`**	**TEI `<bibl/>`**
author	`<name/>`	`<author/>`
orgName	`<collab/>`	`<orgName/>`
editor	`<person-group person-group-type="editor"/>`	`<editor/>`
publisher	`<publisher-name/>`	`<publisher/>`
pubPlace	`<publisher-loc/>`	`<pubPlace/>`
date	`<year/>`	`<date/>`
idno	`<object-id/>`	`<idno/>`
analytic	`<article-title/>`	`<title level="a"/>`
monographic	`<article-title/>`	`<title level="m"/>`
journal	`<source/>`	`<title level="j"/>`
series	`<source/>`	`<title level="s"/>`
unpublished	`<source/>`	`<title level="u"/>`
volume	`<volume/>`	`<biblScope unit="volume"/>`
issue	`<issue/>`	`<biblScope unit="issue"/>`
pages	`<page-range/>`	`<biblScope unit="pages"/>`
chapter	`<chapter-title/>`	`<biblScope unit="chapter"/>`

An interesting observation is that the model is not specific to a particular vocabulary. This means training data can be in one vocabulary while the results of the model are represented in another vocabulary. Also, a mix of vocabularies can be used as training data.

Both the JATS and the TEI vocabularies have specific tags for marking up a bibliographic citation. Both container elements allow character data and do not

impose any ordering restrictions. JATS also offers the non-mixed `<element-citation/>` element which provides more structure. In case of a vocabulary that does impose structure, the mapping can be implemented with XML Conditional Random Fields (XCRFs) [2]. This CRF implementation predicts labels of tree nodes instead of labels for text.

3. The Human Factor

Although the model used performs quite well, it will still make mistakes. Rather than programmatically marking up the recognized citation in the XML document, this paper proposes the treat them as suggestions. A user interface for the SMEs allow them to confirm and correct the citations before marking them up in the XML document. This section describes some of the common errors generated by the model, a user interface to correct them and a mechanism for the system to learn the manually made corrections.

3.1. Analysis of Model Errors

As indicated in the section Section 2.6, the Citation Model has a hard time distinguishing between certain labels. A further statistical analysis on the evaluation errors proves this assumption. Table 3 contains the top 10 errors (about 60% of the total errors):

Table 3. Evaluation Errors

#	**Frequency**	**Actual**	**Expected**
1	129	analytic	monographic
2	60	series	journal
3	45	monographic	analytic
4	37	monographic	O[a]
5	31	analytic	journal
6	29	O	idno
7	18	monographic	journal
8	18	journal	monographic
9	18	author	editor
10	18	journal	author

[a]The *O*utside label of the IOB tagging scheme, indicates absence of a label.

This table reveals three classes of issues:

1. Distinguish between related labels, for example analytic and monographic occur in the same location. This is the cause of results 1, 2, 3, 5, 7, 8, 9, 10.
2. Distinguish whether certain characters are inside a title or not. This is the cause of result 4.
3. Distinguish identifiers, more specifically URLs. This is the cause of result 6.

The first class of issues is caused by labels that are closely related to each other. Intuitively it makes sense that it is hard to determine the difference between article and book titles, even for humans. The range however is correctly determined in all those cases. Meaning, changing the label would suffice for the user to fix the mistake.

The second class of issues is caused by inconsistencies in the evaluation data. Some titles include a dot while it has been omitted from others. This is solved with a post processing rule either consistently placing the dot inside or outside the title. Alternatively, the training data may be updated.

The third class of issues is caused by the model not understanding URLs. This is best solved by adding a URL feature to the model and retraining it.

In conclusion, it is clear that fast majority of issues can be addressed by simply allowing the SME to select a different, but closely related label.

3.2. User Interactions

Based on the analysis in the previous section, we can formulate one requirement: As a SME I can correct the type of title/person in a citation. This requirement is implemented using two UI patterns:

1. A (context) menu options to toggle between the types.
2. A form with dropdown lists to select different types.

The benefit of the first pattern is that it will also work for correcting citations that have been manually put into the system; it works after the fact. Both context menu options and toolbar buttons have been implemented to allow SMEs to use their preferred method. See Figure 1 for an an example of such an UI.

The benefit of the second pattern is that it is task-specific, allowing the SME to work with greater focus. It is implemented in its own dedicated UI component allowing users to quickly validate the recognized results and correct where needed. This puts the SME in control. The dropdowns are populated with the alternatives sorted by the likelihood of the different evaluations of the model. See Figure 2 for an an example of such an UI.

There is no need to choose between the two; they can very well be combined.

3.3. Structured Content Feedback Loop

Integrating ML models in the authoring process helps to automate some of the repetitive tasks that are part of structured content authoring. As shown in the

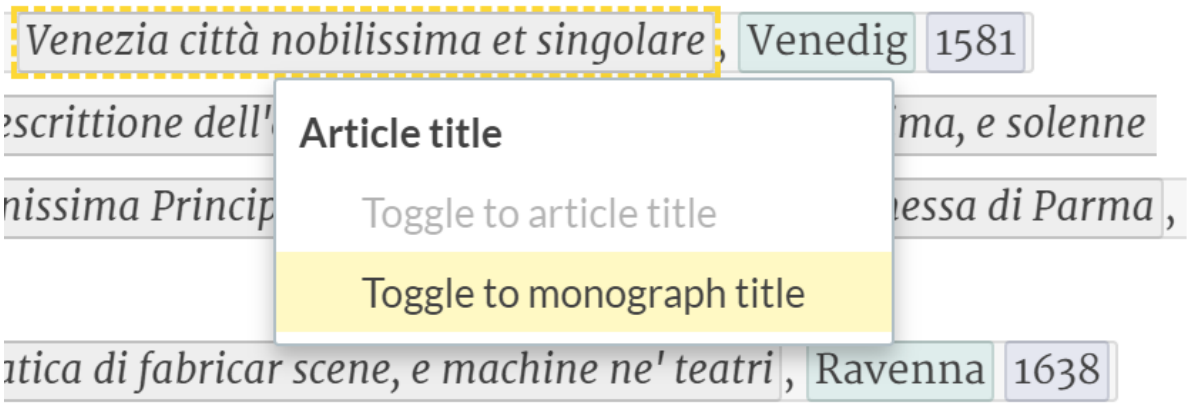

Figure 1. A (context) menu options to toggle between the types

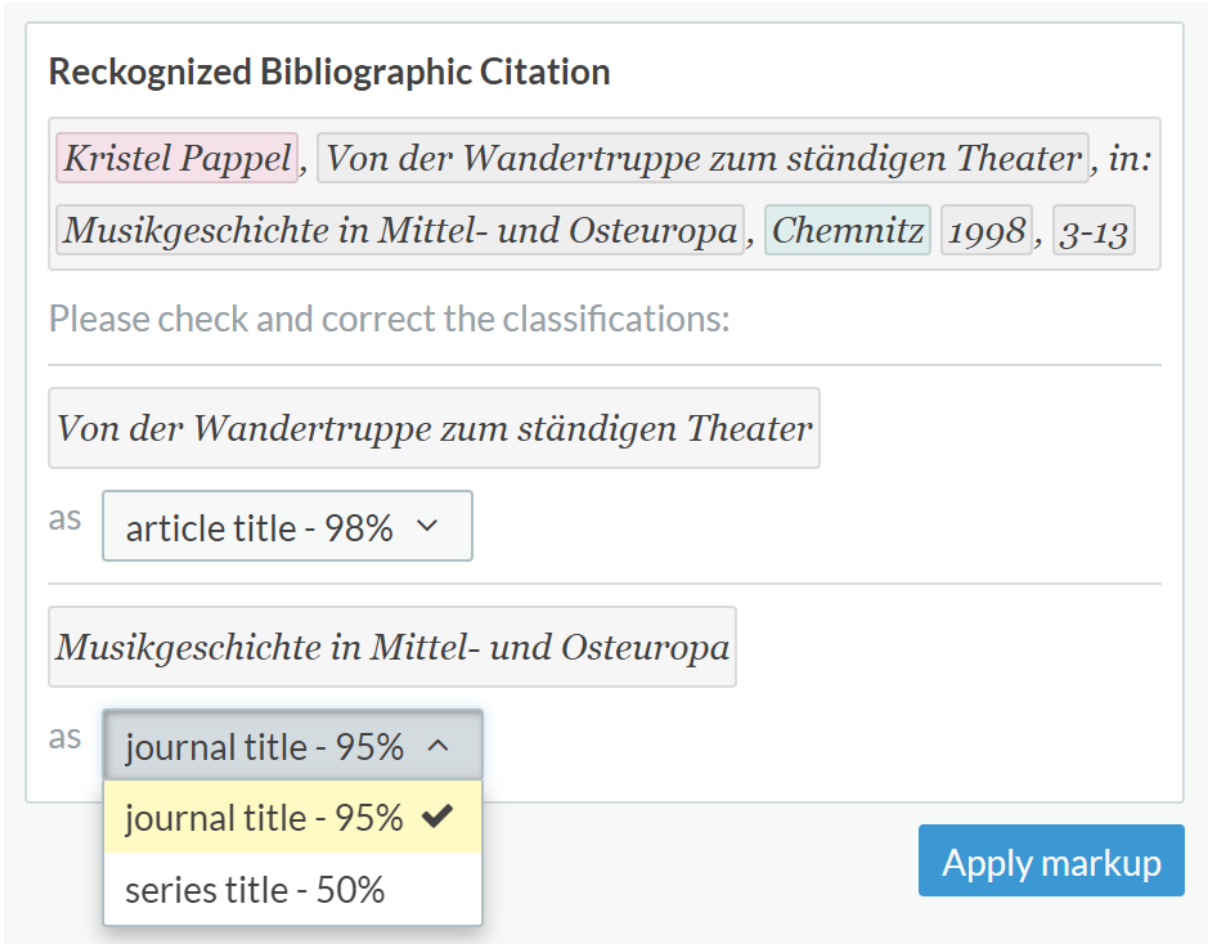

Figure 2. A form with dropdown lists to select different types

analysis of the model performance, there is room for improvement. Integrating the model in the authoring process creates an interesting opportunity to build a feedback loop between the model and the SME.

In order to implement such a loop, the system is expanded with a storage. The storage holds the training data as well as the associated metadata, like date added and user who added the training data. Each record in the training data has a unique identifier. The system is also expanded to include a queue which holds the identifiers of the records that have not been trained on yet. The editor application is modified to send all XML citations to the system upon (auto)save. The system adds the citation to the storage and enqueues it for training. At certain intervals the system retrains the model.

The system has two modes for retraining the model:

1. Incremental retraining; reuses weights from the previously trained model.
2. Full retraining; retrains from scratch.

In both modes the updated corpus (old + new) is used for training. The incremental mode is much faster but has an important limitation in the CRFSharp implementation: it can't handle new labels. The system therefore detects if new labels were added and chooses the appropriate mode. Once retraining has been completed the old model is swapped for the new model and used consecutively for generating the improved suggestions.

Retraining the model automatically imposes some risks. For example, the model may overfit if the SMEs have been working with one particular type of citation for a while. Another example is that an SME (deliberately) inserts the wrong tags throwing the model off guard. The impact of these risks hasn't been explored within the scope of this paper. The metadata, associated with each training record, allows after the fact filtering of records at the cost of a full retraining.

4. Conclusion and Further Work

Using Machine Learning, in this paper Conditional Random Fields, to (semi) automatically markup citations reduces the time SMEs need to spend applying markup manually. In cases where the model mispredicted the correct markup, in general, the correction is just changing the predicted label. Using a specific user interface to change those labels, SMEs stay control while still saving time.

Integrating ML into the authoring process not only saves valuable SME time, it also allows for a Structured Content Feedback Loop to be created. This feedback loop continuously gathers additional training data, reviewed by the SME, to improve the model. The improved model will in turn generate better predictions.

The CRF algorithm works reasonably well but has some limitations. One such limitation is the context window in which it operates, which severely limits its ability to reason on entire sentences. Another limitation is that it requires manual feature engineering. Both limitations can be addressed by implementing a much more powerful model based on bidirectional LSTM in combination with CRFs [13] [4].

Bibliography

[1] Fu, Zhongkai . 2017. CRFSharp. https://github.com/zhongkaifu/CRFSharp.

[2] Gilleron, Rémi, Florent Jousse, Isabelle Tellier, and Marc Tommasi. 2006. "XML Document Transformation with Conditional Random Fields." INEX.

[3] 2008-2017. GROBID. https://github.com/kermitt2/grobid.

[4] Huang, Zhiheng, Wei Xu, and Kai Yu. 2015. "Bidirectional LSTM-CRF Models for Sequence Tagging." CoRR abs/1508.01991.

[5] JATS Standing Committee. 2015. Journal Article Tag Suite. https://jats.nlm.nih.gov/.

[6] Kudo, Taku. 2017. CRF++: Yet Another CRF toolkit. https://taku910.github.io/crfpp/.

[7] Lafferty, John D, McCallum Andrew, and Pereira Fernando. 2001. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data." Proceedings of the International Conference on Machine Learning. 282–289.

[8] Lance A. Ramshaw, Mitchell P. Marcus. 1995. "Text Chunking using Transformation-Based Learning." ACL Third Workshop on Very Large Corpora 82-94.

[9] McCallum, Andrew Kachites. 2002. MALLET: A Machine Learning for Language Toolkit. http://mallet.cs.umass.edu.

[10] Peng, Fuchun, and McCallum Andrew. 2004. "Accurate Information Extraction from Research Papers using Conditional Random Fields." HLT-NAACL.

[11] Sutton, Charles A., and Andrew McCallum. 2012. "An Introduction to Conditional Random Fields." Foundations and Trends in Machine Learning 4: 267-373.

[12] TEI Consortium, eds. 2017. TEI P5: Guidelines for Electronic Text Encoding and Interchange. http://www.tei-c.org/Guidelines/P5/.

[13] Zhiheng, Huang, Xu Wei, and Yu Kay. 2015. "Bidirectional LSTM-CRF Models for Sequence Tagging." CoRR abs/1508.01991.