

Stochastic models for document restructuration

Patrick Gallinari, Guillaume Wisniewski, Ludovic Denoyer, Francis Maes
First_name.name@lip6.fr
Lip 6 — Université Pierre et Marie Curie. Paris- France

Abstract - Document (re)structuration consists in mapping documents coming from different sources, with different formats, onto a predefined semi-structured format. This generic problem appears in different applications settings like heterogeneous semi-structured databases querying, peer to peer systems, legacy document conversion, XML information retrieval. In the paper, we define the restructuration problem from a document centric perspective and identify the main problems raised by this new problematic. We then consider two restructuration instances: structuring flat documents and learning the correspondence between structured formats. We propose stochastic models for these two tasks and describe tests on a large XML document collection.

1. Introduction

With the development and growth of numerical resources, semantically rich data (multimedia, video, text documents, web resources, etc) tend to be encoded using semi-structured formats. Content elements are organized according to some predefined structures which reflects logical or semantic relations between these elements. For instance, XML and, to a lesser extent, HTML allow us to identify elements in a document (like its title or links to other documents) and to describe relations between those elements (e.g. we can identify the author of a specific part of the text). Additional information such as metadata, annotations, etc., is often added to the content description leading to richer descriptions. A key problem for automating the processing of semi-structured resources is the format heterogeneity among data sources. Figure 1 shows the same information (the casting of a movie) captured from two sources, and is a striking example of the problems raised by heterogeneity. For dealing with heterogeneous semi-structured data, the correspondence between the different formats has to be established.

Manual correspondence is usually performed via document transformation languages, like XSLT. However the multiplicity and the rapid growth of information sources have motivated researchers to develop machine learning technologies for helping to automate those transformations. The problem is to map a structured representation onto a reference structure or, more generally, to learn the correspondences between structured representations. This problem has been addressed for some years in the database community and more recently for information retrieval, legacy document conversion, and ontology matching (see the related work section). From a machine learning perspective, different techniques have been used. The problem is often formulated as a classification problem where

Stochastic models for document restructuring

multiple classifiers operating on different data modalities are combined. Learning the correspondence between structured representations is still a new and open problem; depending on the application, many different instances of the problem may be formulated.

We focus here on the matching problem from a document centred perspective (as opposed to database motivated approaches). Potential applications are in the domain of information retrieval like the development of XML search engines for heterogeneous collections ([3], [2]), document filtering ([4]) or legacy document conversion ([1]). More precisely we consider the problem of learning transformations between semi-structured data representations so that heterogeneous documents are mapped onto a predefined semi-structured format. The latter can then be exploited by different applications. This problem is called here the *document restructuring problem*. In the following, we first define this new problem and identify the main difficulties it raises. We then describe two models developed for specific instances of the problem: in the first instance, input documents are supposed semi-structured while, in the second one, we consider the extreme case of flat input documents, the goal being here to learn how to automatically structure plain text. We then describe a series of experiments performed on a large XML document collection.

```
<cast>
  <character>
    <actor>Bruce Willis</actor>
    <characterName>Korben Dallas</characterName>
  </character>
  <character>
    <actor>Milla Jovovich</actor>
    <characterName></characterName>
  </character>
</cast>

<table>
  <tr>
    <td>Korben Dallas</td>
    <td>...</td>
    <td><a href="">Bruce Willis</a></td>
  </tr>
  <tr>
    <td>Leelo</td>
    <td>...</td>
    <td><a href="">Milla Jovovich</a></td>
  </tr>
</table>
```

Fig. 1. Heterogeneity example: two documents describing the same information coming from two different sources. Both the organization, partitioning and element order differ.

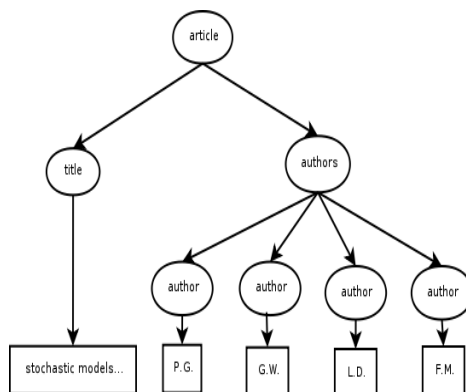


Fig. 2. A tree representation of a semi-structured document. Circle nodes correspond to structural information and square nodes to content information

2. Document Restructuration Problem

We consider that semi-structured data are represented as labelled ordered trees. Inner nodes provide the logical structure, and leaves or attributes provide the content. To each source is associated an alphabet over a set of tag names. The set of trees that may be generated by a given source is supposed to be defined by a schema (e.g. a DTD).

Document restructuring is defined as a tree transformation problem where the semantic of the document is to be preserved. Although this may uncover different situations, we will assume for simplification that all content elements (the *PCDATA* elements) have to be present in the input and output trees. Schemas provide a syntactic definition of the source documents, but do not give any information on the document semantic (the content elements), except the fact that some nodes do contain data. Such information is needed in order to define a transformation. We will assume here that this information is provided by a training set of documents already expressed in the target structure. This set will be used to train the stochastic semi-structured document models which will allow us to perform the mapping task. Output data are assumed to follow a predefined schema, whereas no assumption is made for input data. Indeed, input data are often heterogeneous and may either follow a well-defined DTD, or come from HTML documents or even, sometimes, from plain – unstructured – text documents. Input data may not always conform to a DTD and, often, the latter is not explicitly known.

Formally, let θ be a stochastic document model that has been learned from a training set of target semi-structured documents. We define the restructuring task as the search of the most likely structure d^{optim} according to θ , for a given input document d^{input} :

$$d^{optim} = \arg \max_{d \in \Theta(d^{input})} p(d | d^{input}; \theta) \quad (1)$$

Where $\Theta(d^{input})$ is the set of all possible documents, generated from d^{input} that conform to the target schema. This set should include all possible partitioning and reorganization of the input tree content nodes (node splitting, merging, permutation, etc) and all possible trees corresponding to the new content nodes (or new tree leaves). When an input document is processed, it will be transformed into the most probable structured representation according to the learned target model.

For simplification we do not consider here node splitting or permutation. This means that an input node (e.g. a paragraph) will not be split in the transformation and that the sequence order of content elements is preserved – which is a reasonable hypothesis for textual documents (this is not the case for database-orientated data).

In order to solve this task, one has to learn the parameters θ for a given training set and, then, perform the inference i.e. solve equation 1 for that θ . Note that inference amounts at simultaneously performing the partitioning and labelling of the input document elements, and the target structure generation.

The complexity of computing equation (1) is high because a document can have a lot of possible structures. Under the above hypothesis, if we suppose that both input

Stochastic models for document restructuring

and output document have the same content element sequence (same tree leaves ordering), the number of possible trees that can be built is exponential with respect to the size of this element sequence. That is why, simplifying assumptions will be needed in order to handle large corpora of XML documents.

3. Document Model

Different ML approaches could be developed for solving the above restructuring problem. We propose here to use generative models of semi-structured documents. Other instances of the proposed model have already been successfully used for other tasks such as classification of structured data ([4]). According to equation 1, once a stochastic model is learned, restructuring amounts at finding the most probable structure of an input document.

3.1 Stochastic Generative Models for Semi structured Document

Let d be a tree-structured document as the one represented in figure 2. Let s denote the *structure information* of d , which corresponds to the information conveyed by the XML tree (the nodes, their tag and the relations between them), and c the *content information*. Let θ denote the parameters of a stochastic document model. The probability that a document $d = (s, c)$ has been generated by model θ can be written as:

$$p(d|\theta) = p(c, s|\theta) = p(c|s; \theta) p(s|\theta) \quad (2)$$

The *structural probability* $p(s|\theta)$ describes how the XML tree is generated and the *content probability* $p(c|s; \theta)$ how the model generates the textual information in each leaf node.

With this assumption, we can rewrite equation (1) as:

$$d^{optim} = \arg \max_{d \in \Theta(d^{input})} p(s | d^{input}; \theta) \cdot p(c | s, d^{input}; \theta) \quad (3)$$

Let us now describe these structure and content models.

3.2 Structure Model

The structure model should be able to represent the relations between the different elements of a document. There are, of course, several ways to describe such relations, but usually a compromise must be made between the expressiveness of the model and its complexity (the number of parameters to be learned). We describe below two models which have been developed for two different restructuring tasks.

Stochastic models for document restructuration

3.2.1 Vertical independence assumption –Grammar model

The first model, called here *grammar model*, makes a *vertical independence* assumption, by stating that the probability of an ordered set of children in the tree only depends on their father tag. The structure probability of document d is then given by:

$$P(s | \theta) = \prod_{\text{all internal nodes } n \text{ in } d} P(\text{childrentags}(n) | \text{tag}(n), \theta) \quad (4)$$

where n is a node in the tree, $\text{tag}(n)$ is its tag, and $\text{childrentags}(n)$ is the ordered list of the children tags of node n . This model is closely related to probabilistic context free grammars and the document structure is reduced to a set of relations between a father and the ordered list of its children. The parameters estimated by this model are the probabilities $P(\text{childrentags}(n) | \text{tag}(n))$ and the size of the parameter space is $\text{card}(T \times T^*)$ where T corresponds to the set of possible nodes labels. In the case of XML documents, this space can be very large.

3.2.2 Horizontal independence assumption

The second model called here *Tree Markov Model* (TMM) is similar to the grammar model, but makes use of an additional horizontal independence hypothesis which reduces the parameter space. Children are now supposed generated sequentially according to their parent node tag. Let us denote $n_1^n, \dots, n_{/n/}^n$ the sequence of children of node n where n_i^n corresponds to the i -th child node and $/n/$ is the number of children of n , we make the following assumption:

$$p(\text{childrentags}(n) | \text{tag}(n), \theta) = P(\text{tag}(n_1^n) | \text{tag}(n), \theta) \prod_{i=2}^{/n/} P(\text{tag}(n_i^n) | \text{tag}(n_{i-1}^n), \text{tag}(n), \theta) \quad (5)$$

The size of the parameter space is $\text{card}(T \times T^2)$ which is smaller than the one of the grammar model.

3.2.3 Learning

The two previous models can be learned over a training corpus of semi-structured document using a maximum likelihood algorithm. The learning complexity is linear with respect to the number of nodes of the documents of the training set. More information can be found in [3].

3.3 Content Model

We will assume that the content of a node depends only on the tag of that node -

$$p(c | s) = \prod_{n_i} p(c_i | \text{tag}(n_i)) \quad \text{where } c_i \text{ is the content associated to node } n_i. \quad \text{The}$$

Stochastic models for document restructuring

quantity $p(c_i | \text{tag}(n_i))$ itself can be estimated in different ways depending on the application. In this article, we used simple Naïve Bayes models.

4 Experiments

We performed experiments using the INEX'03 corpus [2]. This is a collection of XML articles from 20 different IEEE journals and proceedings of the IEEE Computer Society. It is composed of about 12,000 documents which represent more than 7,000,000 XML elements. The documents have an average length of more than 7,000 words, 500 segments nodes (nodes containing textual information) and 200 internal nodes (nodes without any content). There are 139 different tags. This is a medium size collection according to IR criteria, but it is quite large for the complex restructuring task.

4.1 Tree labelling structure mapping – grammar model

In a first series of experiments, we considered a simplified version of the restructuring problem where the structure of the tree is kept unchanged so that to each node in the input space will correspond to one node in the output space. Input documents are unlabeled trees (the tree structure of the original document is kept, node tags are unknown). Restructuring then amounts at finding the correct labels for all tree nodes according to INEX DTD.

For this task, consider the input document d^{input} with nodes $n_1, \dots, n_{|d^{input}|}$ where $|d^{input}|$ is the number of nodes of the document. Using equations 3 and 4, we want to find the set of node tags labels $l_1, \dots, l_{|d^{input}|}$ which solves:

$$d^{optim} = \arg \max_{l_1, \dots, l_{|d^{input}|}} \prod_{i=1}^{|d^{input}|} P(l_{i+1}, \dots, l_{i+k} / l_i) P(c_i / l_i) \quad (6)$$

Where $l_{i+1}..l_{i+k}$ is the ordered list of node n_i children labels. Using a Viterbi-like algorithm, the inference complexity is now linear with respect to the number of parameters and to the number of content elements (the leaves in the document tree). In this instance, the restructuring task is simply a multi-label classification task: basically, the content nodes are classified according to the content model (we used here a Naïve Bayes model per tag) and the internal nodes are classified according to the structure model and the content probability of the leaves nodes.

For this experiment we considered three document models derived from the grammar model: a first one using only the content information, a second one using only the structure information and a last model using both kind of information. As a baseline, we also considered a dummy model which labels nodes by using a heuristic

Stochastic models for document restructuring

based on the depth of the node in the tree, *i.e.* by finding $\arg \max_{tag} p(tag | node \text{ depth})$.

All models were evaluated both on the original INEX collection and on a modified version of this corpus where only 5 tags are used instead of the original 139 tags (all 139 original tags were mapped onto these 5 tags). The 5 tags evaluation was motivated by XML IR search engines applications: when querying scientific documents, only a few tags are used in the structured queries (e.g. article, abstract, author, title), so there is no need for restructuring to be able to tackle the whole structure of INEX.

As a performance measure, we used the recall over the document nodes. Recall measures the quantity of correctly labelled nodes. In these experiments, a node is correctly labelled if the restructuring model finds the INEX label (among the original 139 or the simplified 5 ones) of the node.

Recall for the different experiments are shown in table 2 (recall and precision are equal in this case). The full model is able to label correctly about 86% of the nodes for the 5 tags setting and 65% when using 139 tags. Structure information by itself allows us labelling correctly respectively 73% and 49% of the tags and the content information 81% and 38.70%.

The curve in figures 3 gives the ability of the models for tagging correctly full documents. For the 139 tags setting, the full model has correctly tagged almost 90% of the nodes for about 40% of the documents. This is clearly not sufficient for a fully "automatic" labelling. The task is however difficult here since the number of tags is large. In such a situation, the stochastic model could be a component of a more complex labelling system. For the 5 tags experiments, this model has been able to correctly tag about 80% document with an accuracy of about 85%. This level of performance should be acceptable for many IR applications.

Both measures show that structure and content information are complementary and that considering them together significantly increases the performance.

	5 tags	139 tags
<i>Content (1)</i>	58%	27.80%
<i>Content (2)</i>	81%	38.70%
<i>Structure</i>	72.90%	49.70%
<i>Structure + Content</i>	86.50%	65.30 %
<i>Dummy model</i>	79.30%	9.5%

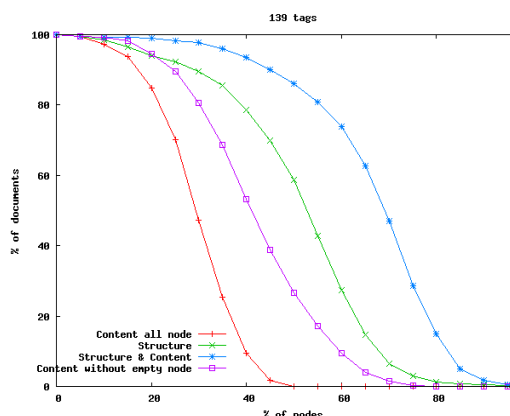


Table 1. Recall for the structure, content and full models for the two experiments. (1) recall is averaged over all nodes, (2) average is performed on

Fig. 3. Percentage of documents with more than % nodes correctly tagged in the tree labelling

Stochastic models for document restructuring

content nodes. only.

experiments (results presented for 139 tags).

4.2 Structure extraction experiments – Tree Markov model

In this second series of experiments, the input is composed of plain text documents. The task then amounts at structuring flat documents according to a predefined schema.

This is an extreme case of the restructuring problem, where the input information is reduced to the document content. In practice this corresponds to loosely structured input documents. The complexity of this task is much more important than for the first experiment and the grammar model has too many parameters to be learned and to be used in practice. The TMM model is more suitable in this case, but even with the additional independence assumptions, the decoding is still too complex. For the experiments we used a suboptimal method where the document structuration is performed in two steps: first the input document is segmented using sequence models; second TMM operates onto the sequence of content segments and outputs a structured document. In this case, the content model is used to find the correct label of each terminal node of the document and the structure model is used to find the non-terminal nodes.

Input documents here are flat versions of the INEX original documents. All structure information, including partitioning and layout (white spaces, tabulations, blank lines, etc.) were removed. Punctuation signs and layout information are used to split the content information into a sequence of *tokens*; punctuation signs are kept as tokens.

Two experiments were conducted: in the first one, the correct segmentation into the original INEX document elements was used; in the second one, the segmentation was performed from the plain document text by ergodic HMMs trained to identify the start and the end of the document elements.

We have used a coverage measure in order to evaluate the models. This coverage measure corresponds to a F_1 measure in the space of the possible nodes items – an item is defined by a triplet $(tag, start, end)$ where tag is the label found, and start and end are the position of the words covered by the node. A description of this measure can be found in [5]. Basically, if the measure is equal to 100%, the model has exactly found the original tree. We show, in table 2, the performance measure for the terminal and for the internal nodes in order to analyze, first, the quality of the segmentation into document elements and, second, the structure inference quality.

An upper bound of the performance with this model is provided by the “Exact” segmentation experiments. Performance drop when the segmentation is inferred from the plain text data. Remember however that this task is an extreme case of restructuring and is extremely difficult. In most cases input documents will have some basic structure which will help during the transformation step. Even for the

Stochastic models for document restructuring

above task, performance could be improved if the labelling and structuration tasks were performed simultaneously, but this is still prohibitive for realistic corpora like INEX.

	Leaf nodes	Internal Nodes
HMM+TMM	33.0 %	29.8 %
Exact+TMM	75.7 %	47.4 %

Table 2. F1 score for the structure extraction experiment – HMM means segmentation with an HMM and Exact, exact segmentation (see text).

5 Related Work

In the database community automatic or semi-automatic data integration — known as *schema matching*— has been a major concern for many years. A recent taxonomy and review of these approaches can be found in [7]. [6] describes one of the most complete approaches which can handle both ontologies, SQL and XML data. The matching task is formulated as a supervised multi-label classification problem. Although there are many similarities with the problem dealt with in this paper, this is a different instance of the restructuring problem. Database data, even XML ones — keep an attribute-value structure, are more regular and have less textual content. [1] is tackling the same problematic than ours in the special case of document conversion (e.g PDF to XML). As for TMMs, they proceed in two steps: element labelling and structure generation.

Document restructuring, also shares many similarities with the information extraction task, which aims at automatically extracting instances of specified classes and/or relations from raw text and more recently from HTML pages. Recent works in this field [11] have also highlighted the need to consider structure information and the influence between extractions.

The document models proposed here also share similarities with other ML models proposed in the literature. Hierarchical HMMs (HHMs, [9]) and Hidden Tree Markov models (HTMM, [8]), also allow to model hierarchical structures like trees and are closely related to the TMM model. HHMMs have two kinds of hidden states: *abstract states* which emit sub-HHMM and *production states* which emit observations (as in HMM). Each abstract state is defined by a *vertical transitions* defining the probability to “activate” the children and by a *horizontal transitions* defining the transition among its children. Both kind of transition also appear in our TMM model.

HTMM also use a *first-order tree-Markov* assumption similar to the one used in TMMs.

Conclusion¹

Matching heterogeneous descriptions of semi-structured data is a key problem in many different domains. We have described here the restructuration problem for semi-structured documents from a document centric perspective and proposed a machine learning formulation of this problem. We have developed a quite general solution to this problem. First stochastic models of target documents are learned from a training set of semi-structured documents. Restructuration then amounts at performing the decoding of input documents according to the learned model. The complexity of this decoding step being high, simplifying assumptions are needed in order to reach a compromise between the models expressiveness and complexity. We have described two instances of generative models which have been developed for two different restructuration tasks. They have been evaluated on the INEX corpus, a large size collection of semi-structured documents. This is work in progress and other ML approaches to this problem (e.g. [10]) have still to be developed.

References

- [1] Chidlovskii, B. and Fuselier J.:, Supervised learning for the legacy document conversion DocEng'04, 220–228, 2004
- [2] Fuhr, N., Govert, N., Kazai, G. and Lalmas, M.:, INEX: Initiative for the Evaluation of XML Retrieval SIGIR Workshop on XML and Informaion Retrieval (2002)
- [3] Denoyer, L., Wisniewski, G. and Gallinari, P.: Document Structure Matching for heterogeneous corpora, SIGIR Workshop on IR and XML
- [4] Denoyer, L. and Gallinari, P.: Bayesian Network Model for Semi-Structured Document Classification Information Processing and Management (2004)
- [5] Johnson M.: PCFG Models of Linguistic Tree Representation, *Computational Linguistic*, 4:613-632, 1998
- [6] Doan, A., Domingos, P. and Halevy, A.: Learning to Match the Schemas of Data Sources: A Multistrategy Approach. *Machine Learning* 50(3): 279–301 (2003)
- [7] Doan, A. and Halevy, A.: Semantic Integration Research in the Database Community: A Brief Survey *AI Magazine*, Special Issue on Semantic Integration, Spring 2005
- [8] Diligenti, M., Frasconi, P. and Gori, M.: Image Document Categorization using Hidden Tree Markov Models and Structured Representations *Proceedings of the International Conference on Applications of Pattern Recognition*
- [9] Fine, S. and Singer, Y. and Tishby, N. The Hierarchical Hidden Markov Model: Analysis and Applications *Machine Learning*, 32(1) 41–62, 1998
- [10] Tsochantaridis, I., Hofmann, T., Joachims, T. and Altun, Ya.: Support vector machine learning for interdependent and structured output spaces, *ICML'04*, 2004
- [11] Freitag, D.: Machine Learning for Information Extraction in Informal Domains, *Machine Learning*, 39, 169–202, 2000.

¹ This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.