Probabilistic Model for Structured Document Mapping

Application to Automatic HTML to XML Conversion

Guillaume Wisniewski, Francis Maes, Ludovic Denoyer, and Patrick Gallinari

LIP6 — University of Paris 6
104 avenue du prsident Kennedy
75015 Paris
name.surname@lip6.fr

Abstract. We address the problem of learning automatically to map heterogeneous semi-structured documents onto a mediated target XML schema. We adopt a machine learning approach where the mapping between input and target documents is learned from a training corpus of documents. We first introduce a general stochastic model of semi structured documents generation and transformation. This model relies on the concept of meta-document which is a latent variable providing a link between input and target documents. It allows us to learn the correspondences when the input documents are expressed in a large variety of schemas. We then detail an instance of the general model for the particular task of HTML to XML conversion. This instance is tested on three different corpora using two different inference methods: a dynamic programming method and an approximate LaSO-based method.

1 Introduction

With the development and growth of numerical resources, semantically rich data tend to be encoded using semi-structured formats. In these formats, content elements are organized according to some structure, that reflects logical, syntactic or semantic relations between them. 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 meta data, annotations, etc., is often added to the content description resulting in richer descriptions.

For many applications, a key problem associated with the widespread of semi-structured resources is heterogeneity: as documents come from different sources, they will have different structures. For instance, in XML document collection focused on a specific domain (like scientific articles), document will come from different sources (e.g. each source corresponds to a journal) and will, therefore, follow different schemas. The schema itself may unknown. For managing or accessing this collection, a correspondence between the different schemas has to be established. The same goes for HTML data on the Web where each site will develop its own presentation. If one wants, for example, to develop a movie database, information has to be extracted from each site so that heteregeneous structures may be mapped onto a predefined mediator schema.

```
<table>
    <tr>                                         <cast>
        <td> Korben Dallas </td>                     <character>
        <td> ... </td>                                   <actor> Bruce Willis </actor>
        <td> <a> Bruce Willis </a> </td>                 <name> Korben Dallas </name>
    </tr>                                            </character>
    <tr>                                             <character>
        <td> Leelo </td>                                 <actor> Milla Jovovich </actor>
        <td> ... </td>                                   <name> Leelo </name>
        <td> <a> Milla Jovovich </a> </td>           </character>
    </tr>                                        </cast>
</table>
```

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

Manual correspondence between heterogeneous schemas or toward a mediated schema is usually performed via document transformation languages, like XSLT. However the multiplicity and the rapid growth of information sources have motivated researchers to work out ways to automate these transformations [1,2]. This heterogeneity problem has been addressed only recently from a content centric perspective for applications in information retrieval [3], legacy document conversion [4], and ontology matching [5]. Depending on the targeted application and on the document sources considered, this semi-structured document mapping problem will take different forms. With heterogeneous XML sources, the correspondence between the different structures will have to handle both the structural and content information. The mapping will provide new structures for the input sources, this is an annotated tree conversion problem which involves tag renaming and document elements reorganization and annotation. For the HTML to XML conversion problem, the context is different. HTML documents are only weakly structured and their format is presentation-oriented. The problem here will be to map this weakly structured visualization oriented format onto a valid XML tree.

In this article, we consider the problem of automatically learning transformations from heterogeneous semi-structured documents onto an XML predefined schema. We adopt a machine learning approach where the transformation is learned directly from examples. We propose a general framework for learning such transformations and focus then on the special case of HTML to XML conversion. The article is organized as follows. The general framework is introduced in Section 2. Section 3 details the HTML to a predefined XML schema conversion problem. Experiments performed on four different corpora are described in Section 4 and related work is reviewed in Section 5.

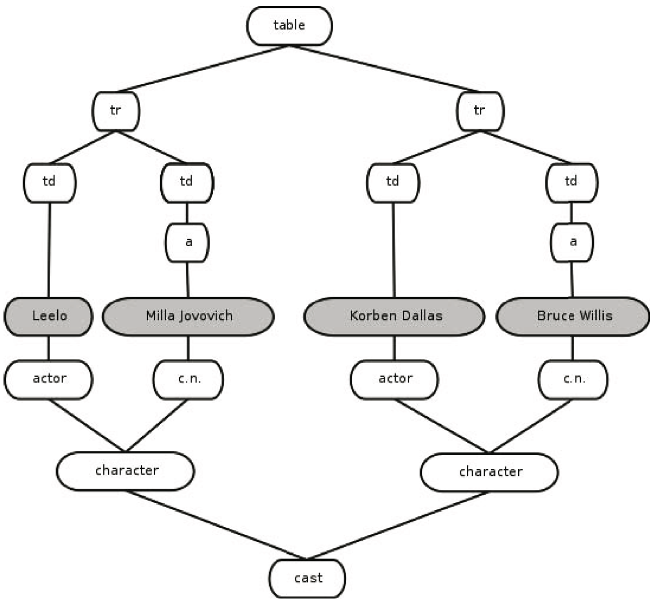

Fig. 2. Toy example of a structured document transformation from HTML data to a predefined schema describing the casting of a movie

2 A Model for Document Structure Mapping

2.1 General Framework

We consider semi-structured documents where content information (text, video, pictures, etc.) is organized according to some schema. In the following, the terms *semi-structured* and *schema* are used in a general sense and are not restricted to XML. The former includes different formats like HTML, XML or PDF and the latter denotes the document organization. We study the problem of learning mappings from a set of heterogeneous documents onto a predefined mediated target XML schema denoted s_T (T holds for Target). The set of possible input schema is denoted $S = \{s_1, ..., s_{|S|}\}$. No assumption is made on the structure of the input documents for the general model. These documents may either follow a well-defined DTD, or may be HTML documents or even plain — unstructured — text documents.

A straightforward approach for learning to map heterogeneous documents onto a target schema is to learn the correspondence for each input schema. This raises different problems: for example representative data have to be collected for each input schema and schemas not represented in the training set cannot be transformed. In order to bypass these limitations, we will introduce an abstract representation of a document, called *meta document*, which will be used as an intermediate representation in our document mapping framework. This abstract representation supposedly contains the information needed for an individual to create the different views of a document corresponding to different schemas. This *meta document* will provide a link between the different representations, it is a variable of our model and its very definition will depend on the precise task we are dealing with. In order to fix the ideas, let us consider an illustration of this concept. In Figure 3, the meta document is represented as a set of relations and content elements which may be stored into a relational database. It may then be used for producing different projections onto different schemas. It may also be transformed into a HTML document for an intranet, into a PDF document or into an XML document following a specific DTD. We denote d_{s_i} the projection of the meta document d onto schema s_i.

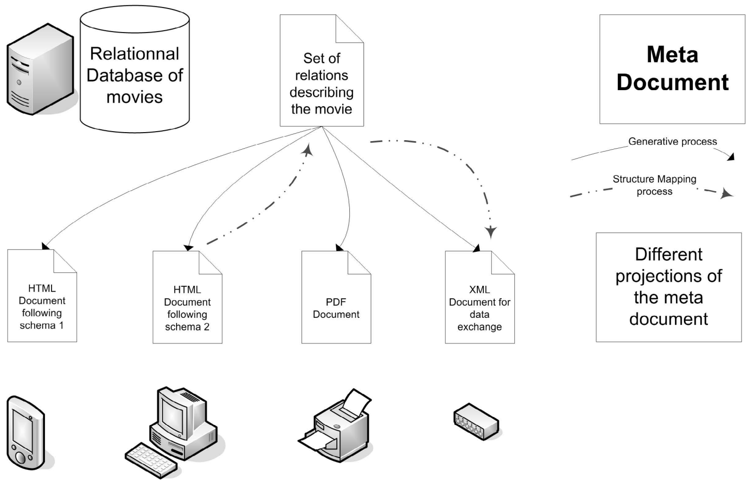
Relationnal Database of movies
Set of relations describing the movie
Meta Document
Generative process
Structure Mapping process
Different projections of the meta document
HTML Document following schema 1
HTML Document following schema 2
PDF Document
XML Document for data exchange

Fig. 3. In this example, a company uses a Database Server to generate different views of a same piece of information of the whole database. Each piece of database describing a particular movie is the *meta document* of the movie.

The meta document d is not necessarily known — in the example of Figure 3, one does not usually have access to the database used to generate the different documents. Different meta documents can produce the same projection onto a schema s_i. For example, different databases can be used to generate the same HTML document. In the proposed model, we will consider that d is a hidden random variable. For the HTML to XML problem dealt with in Section 3, we will propose a specific instance of d.

Our stochastic model of document view generation is described in Figure 4 using a Bayesian network formalism. The meta document d is a latent variable which provides a link between different document representations. a_i is a discrete random variable that represents the author of the projection of d onto d_{s_i} — it identifies the specific process by which d_{s_i} is produced from d. In this model d_{s_i} is fully defined by d and a_i. In practice a_i will simply identify a source. a_T is not represented in this model since the target schema is unique and predefined.

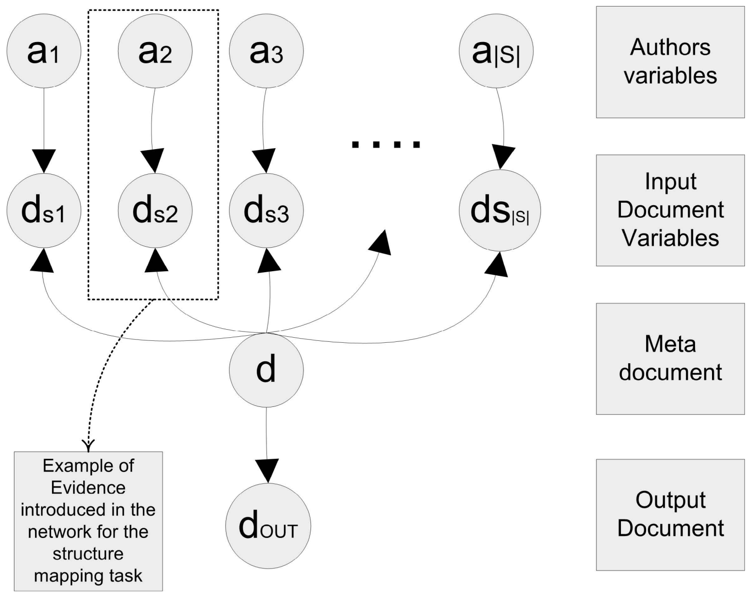

Fig. 4. The belief network representing the generative process of the different views of a meta document

This generative model of document views will serve as a basis for introducing the transformation model.

2.2 Structure Mapping Formalism

We will denote $s_{in(d)}$ the input schema and $d_{s_{in(d)}}$ the projection of d onto this schema. The author of the transformation of d into $s_{in(d)}$ denoted $a_{in(d)}$ may be known or unknown depending on the structure mapping problem. For example, if we want to transform different known Websites into XML documents, the author (the Website) is a known information. Usually there is no information available about the *meta document* d and only a projection of d will be available.

Within this framework, we formalize the mapping problem as follows: given a document $d_{s_{in(d)}}$ and an author $a_{in(d)}$, find the mapping which maximizes the probability of generating a document in the target schema. Formally one could write:

$$d_{s_T} = \underset{d' \in s_T}{\operatorname{argmax}} P(d'|d_{s_{in(d)}}, a_{in(d)}) \tag{1}$$

In order to solve equation 1 we use the document view model of Figure 4. Let us write the joint probability for the whole Bayesian Network:

$$P(d, d_{s_1}, ..., d_{s_{|S|}}, d_{s_T}, a_1, ..., a_{|S|}) = P(d) \prod_{i=1}^{|S|} P(a_i) \prod_{i=1}^{|S|} P(d_{s_i}|d, a_i) P(d_{s_T}|d) \tag{2}$$

Since we only have access to a projection of the document d onto schema $s_{in(d)}$, we will integrate out all unknown variables, leading to:

$$P(d_{s_T}, d_{s_{in(d)}}, a_{in(d)}) = \sum_{\substack{d \\ \{a_k\}_{k \neq in(d)} \\ \{d_{s_j}\}_{j \neq in(d)}}} P(d) \prod_{i=1}^{|S|} P(a_i) \prod_{i=1}^{|S|} P(d_{s_i}|d, a_i) P(d_{s_T}|d) \qquad (3)$$

Here the summation over d consists in integrating over all possible instances of the hidden variable d. From this expression, we obtain the final expression for the right term of Equation 1:

$$P(d_{s_T}|d_{s_{in(d)}}, a_{in(d)}) \propto \sum_d P(d)P(d_{s_T}|d)P(d_{s_{in(d)}}|d, a_{in(d)}) \qquad (4)$$

The structure mapping problem consists in solving the following equation:

$$d_{s_T} = \underset{d' \in s_T}{\operatorname{argmax}} \sum_d P(d)P(d'|d)P(d_{s_{in(d)}}|d, a_{in(d)}) \qquad (5)$$

Here $P(d'|d)$ corresponds to the probability of generating a document into the target schema using the meta document d and $P(d_{s_{in(d)}}|d, a_{in(d)})$ is the probability of generating document $d_{s_{in(d)}}$ according to $a_{in(d)}$. Note that the meta document variable trick allows us to model the processing of heterogeneous databases without having to learn one distinct classifier for each input schema.

Solving equation 5 involves summing over all possible meta-documents d and scoring each possible output d'. In order to efficiently compute the target document probability, we will have to make different simplifying assumptions about the stochastic generation processes corresponding to $P(d_{s_{in(d)}}|d, a_{in(d)})$ and $P(d_{s_T}|d)$. These assumptions will depend on the task and on the type of structure mapping problem. In the following, we will detail these assumptions and the model instance for the HTML to XML conversion task.

3 Model Instance for HTML to XML Conversion

We now focus on learning mappings from heterogeneous HTML sources to a predefined XML schema. In this specific HTML to XML conversion task, we consider one possible input schema (HTML) denoted s_{IN} and different possible authors (for example, "IMDB" and "Allocine" fir the movie corpus - see part Experiments). We will make two assumptions: the first one concerning$P(d_{s_{IN}}|d, a_{d,IN})$ and the second one concerning $P(d'|d)$.

3.1 Meta Document Assumption

Tags in HTML documents are mainly used for the rendering of the document and as such do not provide useful information for the transformation. The latter will be essentially based on the content elements of the HTML document. Since tag names and attributes only bring few relevant information in the case of HTML, in the following, the input for the transformation will be the sequence of the document content elements. This assumption models a deterministic process where a meta document d is built from d_{IN} only keeping the sequence of text segments of the input document.

Formally, for the model described in Section 2, a meta document d will be a sequence of text segments denoted $d = (d^1, ..., d^{|d|})$. Let $(d^1_{IN}, ..., d^{|d_{IN}|}_{IN})$ denote the sequence of segment extracted from d_{IN}, the probability $P(d_{s_{IN}}|d, a_{d,IN})$ is defined as follow:

$$P(d_{s_{IN}}|d, a_{d,IN}) = \begin{cases} 0 \text{ if } (d^1, ..., d^{|d|}) \neq (d^1_{IN}, ..., d^{|d|}_{IN}) \\ 1 \text{ elsewhere} \end{cases} \quad (6)$$

3.2 Target Document Model

We now introduce a target document model which will be used for mapping a meta document representation onto a target schema. Under the above hypothesis, this amounts at inferring the probability of XML trees from a sequence of text segments. This model extends a series of document models already proposed for the classification and clustering of XML documents ([6], [7]).

Let $N_{d_T} = (n_1, ..., n_{|N_{d_T}|})$ denote the set of labeled nodes for an XML document d_T and c_i denote the content of node n_i. If n_i is a leaf node of d_T then c_i will be the content of the leaf, if n_i is an internal node of d_T, c_i will be the content of all the leaf nodes descendant of n_i. Let L_{d_T} denote the set of leaves of d_T, and let $d = (d^1, ..., d^{|d|})$ be a meta document, we have $P(d_T|d) = P(d_T|d^1, ..., d^{|d|})$.

Modeling all structural relations from the target tree would involve a very large probabilistic space for random variable d_T. In our model, simplifying assumptions are made so that structure and content information is represented using the local context of each node of the document. These assumptions have already been successfully tested on the categorization and clustering tasks. We will assume that the label of a node only depends on its content, its left sibling (if any) and its father (if any). With these assumptions, we can write[1] (see Figure 5 for the corresponding belief network):

$$P(d_T|d^1, ..., d^{|d|}) = \prod_{n_i \in L_{d_T}} P(c_i|d^i) \prod_{n_i \in N_{d_T}} P(n_i|c_i, sib(n_i), father(n_i)) \quad (7)$$

where n_i is the label of node i (the XML tag), $father(n_i)$ and $sib(n_i)$ correspond to the label of the father node and the label of the left sibling node of n_i. Remind that c_i is the union of all the content information of children of n_i.

[1] In order to simplify the equation, we don't write $P(c_i|c_j, c_k, ...)$ for the internal nodes. The content of internal nodes are built by a deterministic process so the probability $P(c_i|c_j, c_k, ...)$ is considered to be equal to 1.

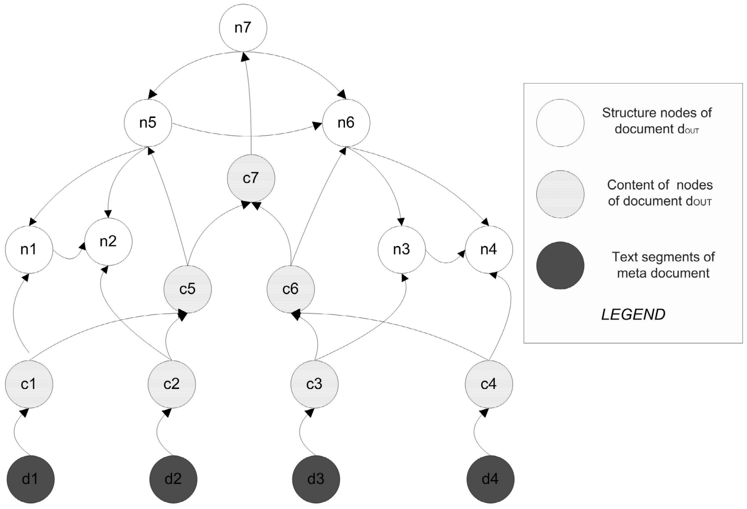
n7
n5
n6
c7
n1
n2
n3
n4
c5
c6
c1
c2
c3
c4
d1
d2
d3
d4
Structure nodes of document dOUT
Content of nodes of document dOUT
Text segments of meta document
LEGEND

Fig. 5. The belief network representing the dependencies between d_T and the sequence of text segments of the meta-document d. The stochastic process modeled here considers that the input content elements d^i generate the content leaf nodes c_i. The label n_i of node i depends on its left sibling, its father and its content.

We make the additional assumption that the leaf content in d_T is exactly the sequence of elements in d (i.e $P(c_i|d^i) = 0$ if $c_i \neq d^i)^2$ which leads to:

$$P(d_T|d^1, ..., d^{|d|}) = \begin{cases} 0 \text{ if } (d^1, ..., d^{|d|}) \neq (c_1, ..., c_{|d|}) \\ \prod\limits_{n_i \in N_{d_T}} P(n_i|c_i, sib(n_i), father(n_i)) \text{ otherwise} \end{cases} \quad (8)$$

Learning the model: In order to learn the probabilities $P(n_i|c_i, sib(ni), father$ $(n_i))$, we have used a maximum entropy framework [8]. The label for each node is chosen by estimating the probability:

$$P(n_i|c_i, sib(n_i), father(n_i)) = \frac{\exp\left(\left\langle W_{n_i}, F_{c_i,sib(n_i),father(n_i)}\right\rangle\right)}{Z_{c_i,sib(n_i),father(n_i)}} \quad (9)$$

where $Z_{c_i,sib(n_i),father(n_i)}$ is a normalizing factor, $F_{c_i,sib(n_i),father(n_i)}$ is a vector representation of the context of node n_i, W_α is the vector of parameters to be learned and $\langle \cdot, \cdot \rangle$ is the scalar product. In this model, we will learn one set of parameters W_α for each possible node label α using a Maximum Entropy method. For the iterative parameter estimation of the Maximum Entropy exponential models, we use one of the quasi Newton methods, namely the Limited Memory BFGS method, which is observed to be more effective than the Generalized Iterative Scaling (GIS) and Improved Iterative Scaling (IIS) for NLP and IE tasks [9].

[2] This assumption corresponds to the idea that we don't want to modify the content of the source document in order to generate the target document.

3.3 Final HTML to XML Model

Once the W_α are learned from a training set, the mapping onto the target schema is finally obtained by solving:

$$d_{s_{FINAL}} = \underset{\substack{d_T \text{ such as} \\ (d^1,\ldots,d^{|d|})=(c_1,\ldots,c_{|d|})}}{\operatorname{argmax}} \prod_{n_i \in N_{d_T}} \frac{\exp\left(\left\langle W_{n_i}, F_{c_i,sib(n_i),father(n_i)}\right\rangle\right)}{Z_{c_i,sib(n_i),father(n_i)}} \quad (10)$$

In order to solve this equation, we have used two methods:

1. The first one is based on dynamic programming (DP) (see [10], [11]) and provides an exact solution to Equation 1. Its complexity is $O(n^3.V)$ (see [10] for more details) — where n is the sequence size of d and V is the number of possible internal node labels - which may be prohibitive for large documents.
2. The second one is based on the LaSO algorithm described in Section 4.1) [12]. It allows us to compute an approximation of the maximum in a complexity of $O(|N_{d_{s_T}}|.V.n)$ where $|N_{d_{s_T}}|$ is the number of node of $|d_{s_T}|$.

4 Experiments

4.1 LaSO-Based Model

LaSO is an original method proposed by [12] that describes a general way to make approximate inference in structure mapping problems. This method is especially useful in cases, like ours, in which dynamic programming is too time-consuming. It relies on the observation that inference can be described as a search process and that it is possible to make it faster by learning an adapted heuristic function and using it in a heuristic search algorithm: this method proposed to consider the learning problem and the decoding problem in an integrated manner.

As we show in the next part, LaSo can be applied very easily to our model and allows us to obtain reasonably good results with a lower inference time but a larger training time.

4.2 Corpora

The HTML to XML structure mapping model has been tested on four different collections. One is the INEX'03 corpus [13], which includes XML articles from 20 different journals and proceedings of the IEEE Computer Society. It is made of about 12,000 documents that represent more than 7,000,000 XML elements. The documents have an average length of 500 leaf nodes and 200 internal nodes. There are 139 tags. This is a medium size collection according to the IR criteria, but it is quite a large corpus for the complex structure mapping task. Each INEX document has a corresponding HTML page extracted from the INEX Website which is the input document.

The second collection includes 10,000 movie descriptions extracted from the IMDb Website[3]. Each movie was represented in both, an XML document created from the relational database and a HTML document extracted from the site. The target XML documents have an average length of 100 leaf nodes and 35 internal nodes labeled with 28 possible tags. The documents have a rather regular structure compared to INEX ones: they have fewer tags and share more structural regularities.

The third collection is a set of 39 Shakespearean plays in XML format[4] converted manually to a simple HTML document. There are only a few documents in this collection, however their average length is huge: 4100 leaf nodes and 850 internal nodes. There are 21 different tags. The main challenge of this collection is related to the length of its documents.

The fourth collection, called Mini-Shakespeare, is the smallest one. As in [10], we have randomly selected 60 Shakespearean scenes from the third collection. These scenes have an average length of 85 leaf nodes and 20 internal nodes over 7 distinct tags.

Each collection was randomly split in two equal parts, one for learning and the other for testing. Due to its complexity, dynamic programming was performed only on documents containing less than 150 leafs – this corresponds to 2200 INEX documents, 4000 IMDb documents –, DP was not applicable at all on the third collection.

4.3 Features and Evaluation Measures

The model uses a sparse vector representation of the context of nodes n_i (F_{n_i} in part 3.3). This vector includes structure and content information. Structure is coded through a set of Boolean variables indicating the presence or absence of a particular $(sib(n_i), father(n_i))$ pair. Content is represented by Boolean and real variables. The former encode layout, punctuation and word presence, while the latter represent the size of the content information (in words) and the different word lengths. This sparse representation generates a large vector space: depending on the corpus, there are often more than one million distinct (structure and content) features.

Our first evaluation measure, *Micro*, is the percentage of correctly annotated leaf nodes. It is similar to the word error ratio used in natural language. Since we are dealing with unbalanced classes (*e.g.* INEX documents are essentially made of paragraphs, so this tag is by far the most frequent), we also use a *Macro* score for leaf nodes: the unweighted average of F1 classification scores of each tag. Internal nodes mapping is measured with the *Internal* metric: it is the F1 score of correctly annotated sub-trees, where a sub-tree is correctly annotated when its tag and its content are both correct[5]. The internal metric is similar to the *non-terminal error ratio* used in [10]. The *Full* metric is a *F1* score on all

[3] http://www.imdb.com

[4] http://metalab.unc.edu/bosak/xml/eg/shaks200.zip

[5] A sub-tree is correctly annotated if its root node has the right label and if its content is **exactly** the target content. This measure is sometimes called *coverage*.

Table 1. Structure mapping results on four XML collections. Four evaluation measures are used (Experiments performed on a standard 3.2Ghz Computer.)

Collection	Method	Micro	Macro	Internal	Full	Learning time	Testing time
INEX	DP	**79.6**%	**47.5**%	51.5%	**70.5**%	30 min	≃ 4 days
	LaSO	75.8%	42.9%	**53.1**%	67.5%	> 1 week	**3h20min**
Movie	DP	**95.3**%	**91.2**%	77.1%	**90.4**%	20 min	≃ 2 days
	LaSO	90.5%	88.6%	**86.8**%	89.6%	> 1 week	**1h15min**
Shakespeare	LaSO	95.3%	78.0%	77.0%	92.2%	≃ 5 days	**30 min**
Mini-shakespeare	DP	**98.7**%	**95.7**%	**94.7**%	**97.9**%	2 min	≃ 1 hour
	LaSO	89.4%	83.9%	63.2%	84.4%	20 min	**1 min**

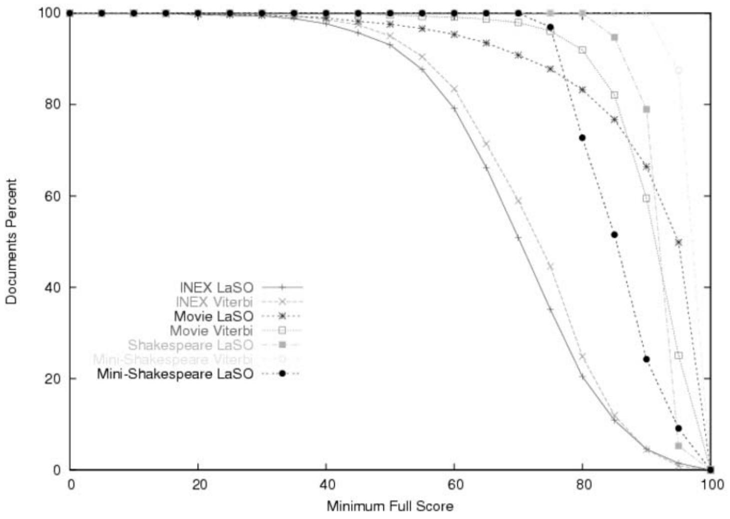

Fig. 6. Percent of documents with more than x% Full score for different values x. We can for example see that the DP method maps correctly more than 80% of the Mini-Shakespeare with a full score included in range [95%, 100%].

built tree components. This is a common measure in the natural language field (under the name of *F1 parsing score*). As a document typically contains more leaf nodes than internal nodes, this measure advantages the leaf score and does not fully inform about the quality of the built tree. These results are shown on Table 1. We also provide the percentage of documents from the test corpus with a Full score greater than than x% (see Figure 6).

4.4 Results

The DP method shows higher scores for leaf nodes classifications than the approximated method based on the LaSO algorithm. For example, with the Movie collection, DP achieves a Micro score of 95.3% whereas LaSO is limited to a score of 90.5%. However, this performance increase has a cost: testing with exact DP inference has a high complexity and may take several days for a collection like INEX, which is unrealistic in practice. It is then limited to short documents. LaSO makes inference fast and practicable for large documents. However, learning is time-consuming. Convergence was not achieved after one week learning on the two real size collections (Movie and INEX). Due to the small number of examples, the huge quantity of features, and the lack of regularization techniques, LaSO also suffers from over-fitting when applied to the Mini-Shakespeare collection.

Best internal scores are achieved by LaSO. This is because LaSO is a top-down parsing method, whereas DP is a bottom-up one. Intuitively, top-down methods may work better on top elements of the trees whereas bottom-up methods are best on bottom elements (leaf nodes).

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 [5]. [14] describes one of the most complete approach which can handle both ontologies, SQL and XML data.

The matching task is formulated as a supervised multi-label classification problem. While many ideas of the database community can be helpful, their corpora are completely different from the textual corpora used in the IR community: all documents — even XML ones — keep an attribute-value structure like for relational database and are thus much smaller and more regular than for textual documents; textual data hardly appears in those corpora. With database corpora, finding the label of a piece of information is enough to build the corresponding tree because each element usually appears once in the tree structure.

Document structure mapping, also shares 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 [15] have also highlighted the need to consider structure information and relations between extracted fields.

The document model proposed here is related to other ML models of the literature. Different authors ([16], [10]) have proposed to use natural language formalisms like probabilistic context free grammars (PCFG) to describe the internal structure of documents. Early experiments [11] showed that the complexity of tree building algorithms is so high that they cannot be used on large corpora like INEX. Our specific XML model makes the same kind of independence assumptions as Hierarchical HMMs ([17]) do. The work closest to ours is [10]. They address the HTML to XML document conversion problem. They make use of PCFGs for parsing text segments sequences of and of a maximum entropy classifier for assigning tags to segments.

6 Conclusion

We have proposed a general framework for the structure mapping task on heterogeneous corpora. Our model uses a *meta document* abstraction in order to generate different views of the same document on different schemas and formats. We have then detailed a specific application of this model for the mapping of HTML document onto a mediated XML schema. From our knowledge, this model is today the only one able to handle large amount of documents for the HTML decoding task. For this problem, the *meta document* is a sequence of text segments and the model will find the best XML tree in the target schema. This model has been implemented using two inference methods: a DP exact method and an approximate LaSO algorithm. The results show that, for both methods, the model is able to cope with large corpora of documents. LaSO is faster than DP and this type of method should be investigated further for the transformation problem.

Acknowledgments

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.

References

1. Chung, C.Y., Gertz, M., Sundaresan, N.: Reverse engineering for web data: From visual to semantic structures. In: ICDE (2002)
2. Zhang, S., Dyreson, C.: Polymorphic xml restructuring. In: IIWeb'06: Workshop on Information Integration on the Web (2006)
3. Wisniewski, G., Gallinari, P.: From layout to semantic: a reranking model for mapping web documents to mediated xml representations. In: Proceedings of the 8th RIAO International Conference on Large-Scale Semantic Access to Content (2007)
4. Chidlovskii, B., Fuselier, J.: Supervised learning for the legacy document conversion. In: DocEng '04: Proceedings of the 2004 ACM symposium on Document engineering, New York, NY, USA, pp. 220–228. ACM Press, New York (2004)
5. Doan, A., Halevy, A.: Semantic integration research in the database community: A brief survey. AI Magazine, Special Issue on Semantic Integration (2005)
6. Denoyer, L., Gallinari, P.: Bayesian network model for semi-structured document classification. Information Processing and Management (2004)
7. Denoyer, L.: Xml document mining challenge. Technical report, LIP6 (2005)
8. Berger, A.L., Pietra, V.J.D., Pietra, S.A.D.: A maximum entropy approach to natural language processing. Comput. Linguist. 22, 39–71 (1996)
9. Malouf, R.: A comparison of algorithms for maximum entropy parameter estimation. In: COLING-02. proceeding of the 6th conference on Natural language learning, Morristown, NJ, USA, pp. 1–7. Association for Computational Linguistics (2002)
10. Chidlovskii, B., Fuselier, J.: A Probabilistic Learning Method for XML Annotation of Documents. In: IJCAI (2005)

11. Denoyer, L., Wisniewski, G., Gallinari, P.: Document structure matching for heterogeneous corpora. In: Workshop SIGIR 2004. Workshop on IR and XML, Sheffield (2004)
12. Daumé III, H., Marcu, D.: Learning as search optimization: approximate large margin methods for structured prediction. In: ICML '05. Proceedings of the 22nd international conference on Machine learning, New York, NY, USA, pp. 169–176. ACM Press, New York (2005)
13. Fuhr, N., Govert, N., Kazai, G., Lalmas, M.: Inex: Initiative for the evaluation of xml retrieval. In: SIGIR'02 Workshop on XML and Information Retrieval (2002)
14. Doan, A., Domingos, P., Halevy, A.: Learning to match the schemas of data sources: A multistrategy approach. Mach. Learn. 50, 279–301 (2003)
15. McCallum, A.: Information extraction: distilling structured data from unstructured text. Queue 3, 48–57 (2005)
16. Young-Lai, M., Tompa, F.W.: Stochastic grammatical inference of text database structure. Machine Learning (2000)
17. Fine, S., Singer, Y., Tishby, N.: The hierarchical hidden markov model: Analysis and applications. Machine Learning 32, 41–62 (1998)