From Layout to Semantic: A Reranking Model for Mapping Web Documents to Mediated XML Representations

Guillaume Wisniewski & Patrick Gallinari

LIP6 — UPMC

104 avenue du prsident Kennedy

75016 Paris

France

`{guillaume.wisniewski,patrick.gallinari}@lip6.fr`

Abstract

Many documents on the Web are formated in a weakly structured format. Because of their weak semantic and because of the heterogeneity of their formats, the information conveyed by their structure cannot be directly exploited. We consider here the conversion of such documents into a predefined mediated semi-structured format which will be more amenable to automatic processing of the document content. We develop a machine learning approach to this conversion problem where the transformation is learned automatically from a set of document examples manually transformed into the target structure. Our method proceeds in three steps. Given an input document, document elements are first annotated with labels of the target schema. Structured candidate documents are then generated using a generalized probabilistic context-free parsing algorithm. Finally candidates are reranked using a perceptron like ranking algorithm. Experiments performed on two different datasets show that the proposed method performs well in different contexts.

1 Introduction

Semantic technologies [Liu and Chen-Chuan-Chang, 2004, Doan et al., 2003] have improved results in several information access tasks, like browsing [Buyukkokten et al., 2001], searching [Kazai et al., 2003] or editing the Web. This has been achieved by considering a semantically-rich structured representation of documents, which allows applications to access both meaningful *content elements* of a document and *relations* reflecting logical or semantic links between them. In the case of Web documents, the classical flat document representation or the loosely structured representation inferred from the markup (the HTML tags) offers only limited descriptive power and cannot be easily processed by semantic technologies. There is hence a need to augment these documents and express them in a format which is more amenable to automatic processing by Semantic Web applications.

Some work [Aumueller, 2005] has already started adapting information organization and production in order to encourage content providers to publish their data in semantic-oriented formats. Besides these initiatives, different tools help authors to enrich flat or HTML documents by manual annotations. This strategy is well adapted for the creation of document collections aiming at professional use and for a specific domain like law or medicine. Note that collective initiatives like Wikipedia [Völkel et al., 2006] have also started to propose tools to enrich documents produced by collaborative authors.

On the other hand, the vast majority of documents is — and probably will remain — accessible in a presentation-oriented format only. This is the case for most Web documents which are written in plain HTML, but also of many *legacy* documents owned by companies and organizations [Chidlovskii and Fuselier, 2004]. These documents have been designed for readability and not for automatic processing. In order to make the content of this enormous stock of documents processable by various applications, some authors have recently started to develop automatic and semi-automatic tools for the conversion of unstructured or loosely structured documents

into a structured format [Chidlovskii and Fuselier, 2005, Mukherjee et al., 2003, Ishitani, 2003]. Although there are different instances of this conversion problem, most approaches aim at converting documents into a predefined semi-structured format which complies with a schema. This schema, defined a priori, encodes the syntax and semantics of the structure needed by a target application.

We address here the problem of automatically transforming HTML documents into a predefined XML format. This is an important instance of the conversion problem introduced above, since plain HTML documents are the most common format on the Web. Our hypothesis is that this conversion is made possible by taking advantage of the structure defined by the DOM tree of HTML document: even though this structure describes only the layout of the document, we believe that the information it conveys enables the extraction of both individual data elements and semantic relationships among them. The layout actually organizes the document in a way that facilitates human browsing and information access: it emphasizes elements (like a title or the price of an item) and relations between those elements. As [Mukherjee et al., 2003] stresses, this relation between semantic and layout has been reinforced by the development of content management systems: today, most Web pages are no longer hand-crafted, but generated from relational databases and their design mirrors the logical structure of documents. Striking examples are the thread organization of user-provided comments in news portal (Figure 2) or the chronological order of the entries of a blog.

In this work, we consider HTML documents as *primary sources* from which structured "views" should be extracted in order to meet semantic Web application requirements. Figure 1 illustrates this conversion process from a loosely structured document representation to an XML document. This process requires both labeling the semantic role of elements and organizing them hierarchically in a way that reflects the desired structure. In this problem, the target schema is assumed to convey structural and semantic knowledge which will be used by different applications. Additional semantic information could also be added via the use of semantic resources like thesauri or ontologies.

The proposed approach proceeds in three steps. First relevant elements of input HTML pages are extracted and annotated by means of a conditional Markovian sequence model. In a second step, annotated document elements are organized into tree structures that comply with the target schema. This structure generation step is performed by a parsing algorithm. At the end of this second step, several candidate tree representations of a document have been generated. In a third step, the "best" candidate is selected by a machine learning ranking algorithm. This three step process provides several improvements over existing approaches like [Viola and Narasimhan, 2005] or [Chidlovskii and Fuselier, 2005]. First it offers a computationally feasible approximate solution to the problem of structure extraction from HTML documents. Second, it allows us to exploit local and global features defined over the input and output documents (especially features describing the information conveyed by the layout). Using both local and global features considerably increases the performance compared to most other systems which rely only on local features. Finally, this approach ignores elements which are not relevant for the target schema in the transformation process.

This article is organized as follows: In Section2 we introduce the general framework; Section 3 details our model and Section 4 the features we used; related work is reviewed in Section 5 and experiments performed on different corpora are described in Section 6.
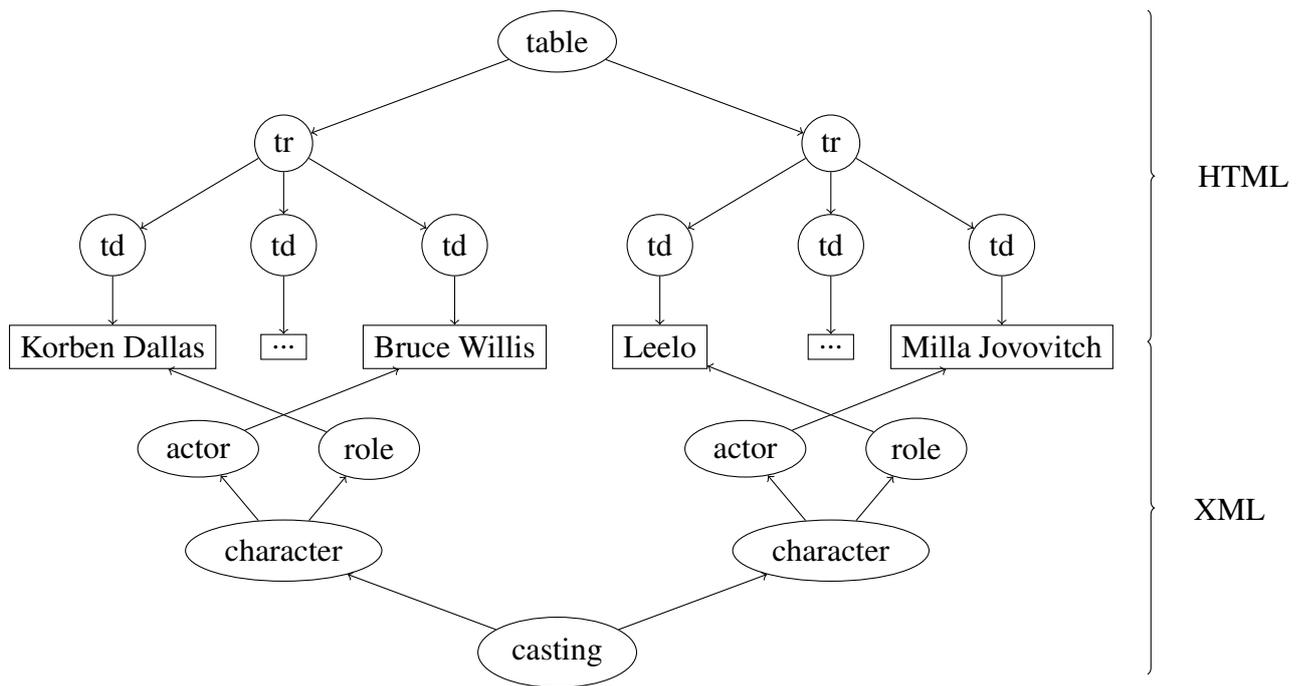
Figure 1: Toy example of a structured document transformation: the aim here is to annotate HTML content elements (Korben Dallas is annotated with role and Bruce Willis with actor) and to map these nodes onto the target XML schema, by identifying which actor plays which part.

Figure 2: Excerpt of a thread of user-provided comments on "Slashdot", a popular technology-related Website. The layout allows us to easily identify comments and relations between them. Meta-information, like the author, or the date the comment has been published are also immediately accessible.

2 Learning for Structure Mapping

The general problem we consider here is the following: We want to transform an input document d^{in} into an XML representation, t^{out}, suitable for a target application. Elements, annotation tags and relations between document elements are defined by the *target schema* specific to the application on hand. Figure 1 illustrates this process on a toy example. Here, the target schema defines a representation for movie castings and simple relations between document elements. The input document contains a table structure from which both document elements (actors and roles) and the relations between elements (which actor plays which part) shall be annotated and mapped onto the target XML structure.

In order to perform this transformation, we propose to learn a mapping from the input document d^{in} to the target structured output t^{out}. We assume that there is a set of HTML documents together with their corresponding XML target documents. We call this set the training set and denote it $S = \left(d_i^{\text{in}}, t_i^{\text{out}}\right)_{i=1}^{N}$. The mapping will be learned from this set of examples. Given an input HTML document and a target schema, different transformations of the input document are possible: In our example, any actor can, in principle, play any role. Let us denote $\mathcal{T}(d^{\text{in}})$, the set of documents resulting from all possible transformations of d^{in} that comply with the target schema. We define a θ-parametrized score function $F(d^{\text{in}}, t; \theta)$ that measures the *degree of compatibility* between the input document d^{in} and each transformed document t of $\mathcal{T}(d^{\text{in}})$. F allows us to rank the elements of $\mathcal{T}(d^{\text{in}})$. Thus the structure mapping task amounts to finding the most compatible candidate:

$$t^{out} = \underset{t \in \mathcal{T}(d^{\text{in}})}{\operatorname{argmax}} F(d^{\text{in}}, t; \theta) \qquad (1)$$

The score function (its parameters θ) is estimated from the training set S. From a machine learning perspective, this formulation of the problem corresponds to the structured learning framework recently proposed by [Tsochantaridis et al., 2004], which generalizes multiclass learning to the case of interdependent and structured labels like sequences or trees.

In Equation 1, the `argmax` operator represents a search through the space of all valid transformations $\mathcal{T}(d^{\text{in}})$. This space includes all possible partitionings and reorganizations of the input content nodes (node splitting, merging, permutation, etc.) and all possible trees which correspond to this new organization. This search space grows exponentially with the number of content elements: for instance, if only node permutations can occur, we already have to consider a search space of size $n!$.

To find the best solution without developing an explosive combinatorial search, several sources of information must be considered: the content of the source document, its structure, the definition of the target structure and its regularities. Indeed, as pointed out by [Viola and Narasimhan, 2005], dependencies between target labels of elements are very useful to disambiguate solutions. For instance, in the example of Figure 1, knowing that `Bruce Willis` is an `actor` indicates that candidate solutions in which `Korben Dallas` is labeled as a `role` should have an increased score. In our case, knowing the target schema facilitates the use of this kind of information: the schema constrains the output by specifying extensively the legal combinations of labels

Another valuable source of information is provided by the input structure. For example, the segmentation of the input content will help extracting the information relevant for the target document. Moreover some relations among elements in the input structure can be exploited for selecting the right structure among several potential output candidates. With regard to the semantic of the document, the structure of the input document helps disambiguating between valid

structured documents and a structure mapping model should aim at uncovering and exploiting these dependencies.

More precisely, three kinds of dependencies can be used to help in computing the structured output document:

- local dependencies, i.e. between neighboring labels;

- long-distance dependencies, i.e. dependencies between two labels which might be arbitrarily distant from each other;

- global dependencies or constraints: For example, one can define features that rely on observations like "a blog entry can only have one date of publication". Other features will encode the proximity of elements in the input document.

One contribution of this work is to present a general machine learning system that can take advantage of these three kinds of features. It has long been noticed [Collins and Koo, 2005, Sutton and McCallum, 2004] that using global features and long-term dependencies is critical to the performance of many natural language processing or information extraction systems. However, the introduction of non-local features usually results in a prohibitive algorithmic complexity: Finding the best target structure to a given input document d^{in} usually relies on techniques based on dynamic programming. These can only be applied if the score function can be factorized in a product of functions depending only on local features.

The proposed method, based on the idea of *reranking*, offers a low complexity solution to the transformation task and can therefore be applied to large corpora.

3 A Reranking Model for Structure Mapping

3.1 General Principle

Building structured representations from loosely structured data generally leads to a problem of exponential complexity. The search space of potential candidate structures is most often combinatorial. Reranking [Collins and Koo, 2005] is a general strategy which offers approximate solutions to such combinatorial problems. This method separates the generation of candidates from the selection of the best one. In a first step, potential candidates are computed. Usually only a limited set of candidates will be generated. Secondly a search among these potential candidates will allow to select the "best" one. This method has been successfully used for tasks like syntactic analysis [Collins and Koo, 2005], in which the goal is to build a tree structured representation from a sequence.

In our case, this strategy has an additional advantage. It allows us to use different representations of the documents at each step. The generation of candidate solutions is performed via dynamic programming techniques which can only take into account local characteristics of the documents. Global features and long distance dependencies can then be exploited during the ranking step which selects the best solution from the potential ones.

In order to implement this approach, we decompose our mapping problem into three successive steps:

1. An *annotation step* in which relevant nodes from the input document are annotated with labels defined by the target schema. As detailed in Paragraph 3.2, the annotation task corresponds to a sequence labeling task generating the set of L best candidates.

2. A *tree-generation* step, described in paragraph 3.3, extracts relations between elements. This is achieved by first checking whether the labeled sequences generated by the annotation step comply with the target schema constraints enforced. If so, candidate trees are generated, otherwise the next best solution in the annotation step is considered.

Checking the compliance of the labeled sequences with the schema introduces a feedback loop between the annotation and tree-generation step. The first two steps *generate* a set $\text{GEN}_N(d^{\text{in}})$ containing the N candidate solutions which consider only local features and dependencies. The overall complexity of the generation step is in $\mathcal{O}(N \cdot n^3)$ where n is the number of relevant elements extracted in the annotation step.

3. A *ranking step* which finds the best solution among all candidates in $\text{GEN}_N(d^{\text{in}})$. Global features and relations between the target and the source documents are considered at this step.

N is a preset parameter of the algorithm that controls the size of the exploration space and therefore the trade off between computational complexity and accuracy.

The whole process is summarized in Figure 3 and the three steps are detailed in the next sections.

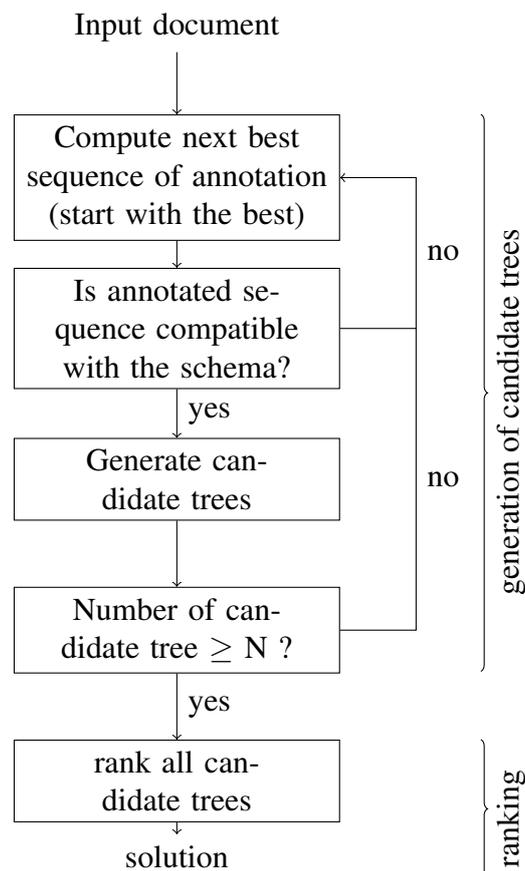Figure 3: Principle of the reranking approach: given an input document the generation step builds a set of candidate solutions; a ranking step then chooses the best solution among these candidates. The generation of candidate solutions consists in two steps: sequence labeling and checking that candidate solutions comply with the target schema. A feedback loop insures that, at least, N candidate solutions are built.

3.2 Annotation Step

In the first step of the transformation process, we label all leaves of the input documents. The set of possible labels is defined by the target schema: Labels of the leaves correspond to the simple elements of the schema. An additional "`ignore`" label is introduced to indicate elements that have to be dismissed. This labeling exploits the information conveyed by the input structure: Elements to extract are generally highlighted by the layout and are consequently stored in separate elements. In the following, we will assume that no further segmentation of the input data has to be performed.

We used linear conditional random fields (CRF) [Lafferty et al., 2001] for this step. CRFs provide an extensible and flexible model and have achieved state-of-the-art performances in various segmenting and sequence labeling task (part-of-speech tagging, information extraction, text chunking, ...).

The probability of generating a particular label sequence $\mathbf{y}$ given a sequence of leaves $\mathbf{c}$ is defined by:

$$p(\mathbf{y}|\mathbf{c};\theta) = \frac{1}{Z(\mathbf{c})} \exp\left(\sum_{t=1}^{n}\sum_{i} \theta_i \cdot g_i(y_{t-1}, y_t, \mathbf{c})\right) \qquad (2)$$

in this expression $Z(\mathbf{c})$ is a normalization factor, n is the length of the sequence, $g_i(y_{t-1}, y_t, \mathbf{c})$ is a *feature function* and the θ_i are parameters to be learned. Each feature function g_i is an arbitrary function of its arguments. The number of functions is arbitrary. CRFs make a first order Markov assumption on the label sequence (only local dependencies among two successive labels (y_t, y_{t-1}) can be considered). Otherwise, the feature functions may depend on global characteristics (even dependent and overlapping ones) over the input sequence. In practice, the g_is are often simple binary functions of their arguments, but any real valued function could be used. Features used for our labeling problem will be detailed in Section 4.

The most likely label sequence can be found efficiently using the Viterbi algorithm [Lafferty et al., 2001]. This algorithm can be extended to build the N-best labeling [Jurafsky and Martin, 2000]. In our experiments, we used the FlexCRF toolkit [Phan and Nguyen, 2005].

3.3 Generation of Candidate Trees

This step aims at checking whether the labeled sequences generated in the previous step comply with the target schema and, if it is the case, at building corresponding candidate trees. These two actions can be performed by modeling schema with Extented Context Free Grammars (ECFG) and by using *parsing algorithm*. Formal grammars are a natural way to model tree structures and especially recursive structures as the ones that are found in Web documents. Several works [Young-Lai and Tompa, 2000, Chidlovskii and Fuselier, 2005, Viola and Narasimhan, 2005] have highlighted the link between schema (especially DTD and XML Schema) and Extented Context Free Grammars. We briefly introduce below ECFGs and explain how it can be used to generate candidate trees.

An ECFG [Jurafsky and Martin, 2000] consists of a set of terminal symbols N, a set of non-terminal symbols Σ, a designated start symbol S and a set of rewriting rules or productions R. Each production is of the form $\Sigma \mapsto (\Sigma \cup N)^*$ and describes how the elements of Σ can be rewritten or *expanded* in a sequence of elements of Σ and N. In the following, we call *string* a sequence of terminals. For readability and succinctness, the rules are generally expressed using regular expressions. For efficiency reasons, ECFGs are converted into a more standardized form: the Chomsky Normal Form (CNF).

ECFGs generate strings that comply with the constraints enforced by the grammar. A string is generated by, starting from the start symbol, successive applications of production rules (any number of times and in any order). The sequence of rule expansions is commonly represented by a parse tree. Efficient parsing algorithms, like the Cocke-Younger-Kasami (CYK) algorithm [Jurafsky and Martin, 2000], determine whether a string can be generated by a given CNF grammar and, if so, how this is done. The complexity of CYK is in $\mathcal{O}(n^3 \cdot \#R)$, where n is the length of the input sequence and $\#R$ the number of productions in the grammar.

Given a grammar, there are generally multiple different ways of generating a single string, especially for the highly recursive structure of Web documents. This is known as the *ambiguity* issue in parsing. That is why, parsing algorithms result in a forest of possible parse trees.
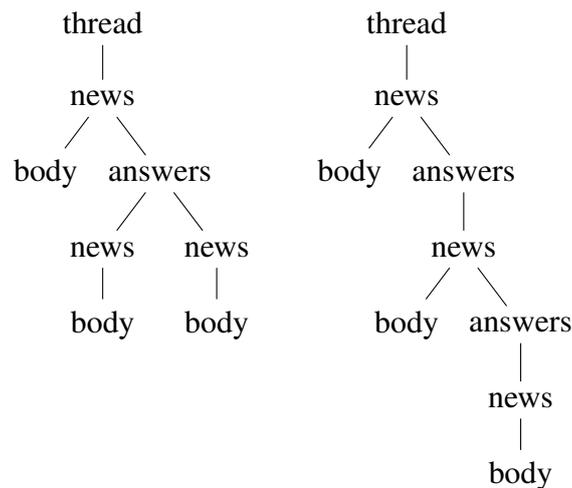
Figure 4: Example of two trees that comply with the schema of Figure 5 and that can be generated by the sequence of terminals body body body.

As shown in Figure5[1], a given schema can be directly transformed into the corresponding ECFG:

- the set of terminal symbols corresponds to simple elements defined by the schema (i.e.: elements that contain only text and no nested elements);

- the non-terminal symbols correspond to the complex elements defined by the schema;

- the productions correspond to the constraints imposed by the schema

It is therefore possible to use a parsing algorithm to check whether a candidate labeled sequence complies with the target schema. The set of candidate trees associated with this sequence is then defined as the result of the parsing of this sequence.

Generally speaking, the number of parse trees that can be generated from a string cannot be known in advance. That is why, we introduced a feedback loop in the process, in order to generate as many candidate strings as needed to get at least N trees at the end of the generation step.

---

[1]For succinctness, the schema is presented as a DTD, but in our experiments we considered schema expressed in XML Schema

```
<!ELEMENT thread (comment)*>
<!ELEMENT comment ((body, answers)|body)>
<!ELEMENT answers (comment)+>
<!ELEMENT body (#PCDATA)>
```
(a) Schema
```
THREAD → COMMENTS*
COMMENT → body ANSWERS
COMMENT → body
ANSWERS → COMMENTS*
```
(b) Corresponding grammar

Figure 5: A schema and the corresponding grammar. This schema describes a thread of user provided comments in a news portal.

3.4 Ranking Step

The final ranking step aims at computing, from the set of candidate solutions, the most plausible output structure corresponding to the input document. To achieve this, a θ-parametrized function $F(t, d^{\text{in}}; \theta)$ is learned and then used to score all candidate trees and find the most plausible one.

3.4.1 Learning

The parameter vector θ is estimated from a set of n training examples. The i'th element of the training set consists in:

- a pair $\left(d_i^{\text{in}}, t_i^{\text{out}}\right)$ of an input document and the corresponding expected output

- a set of candidate solutions $\text{GEN}_N(d^{\text{in}}_i) = (t_{ij})_{j=1}^{n_i}$. This set is the output of the generative process described in the previous section. We assume, without loss of generality, that t_{i1} is the correct output (i.e.: $t_{i1} = t^{\text{out}}_i$).

As a training method, we use a kernel-perceptron [Cristianini and Shawe-Taylor, 2000]. The perceptron is a simple, yet efficient algorithm. As described by Algorithm 1 it goes over the training set several times updating the parameter vector each time an error is made, i.e. each time t_{i1} has not the highest score. The perceptron is then trained to rank the desired output for any document of the training set for this input document.

The score function, which measures the compatibility of a candidate solution with the input document, is defined by:

$$F(t, d^{\text{in}}; \theta) = \sum_{i,j} \theta_{ij} \cdot \left(k\left((t_{i1}, d_i^{\text{in}}), (t, d^{\text{in}})\right) - k\left((t_{ij}, d_i^{\text{in}}), (t, d^{\text{in}})\right) \right) \tag{3}$$

where the sum goes over all the input documents d_i^{in} in the training set and all corresponding candidate solutions t_{ij}. The θ_{ij} are parameters to be learned. k is a kernel function that measures the similarity between its two arguments. It will be precisely defined in Section 4. Using a kernel allows us to consider any feature as long as an efficient procedure to compute the inner product of two feature vectors is available. It is thus possible to represent data in a high dimensional features space, which is well-suited for describing long-term dependencies.

Algorithm 1 Learning algorithm for the kernel perceptron

---

n is the number of examples
n_i is the number of candidate solutions for the i'th example
$\forall i \in [\![1, n]\!] \forall j \in [\![1, n_i]\!], \theta_{ij} = 0$
$F(t, d^{\text{in}}; \theta)$ is defined by Equation 3
while there are ranking errors **do**
    for $i = 1$ to n **do**
        $j = \text{argmax}_{j \in [\![1, n_i]\!]} F(t_{ij})$
        if $j \neq 1$ **then**
            $\theta_{ij} = \theta_{ij} + 1$
        end if
    end for
end while

3.4.2 Inference

Once the parameters θ_{ij} have been learned, when a new document d^{in} is processed, the score $F(t, d^{\text{in}}; \theta)$ of all candidates is computed and the one with the highest score is selected as the proposed solution.

4 Features

As explained in Section 2, several sources of information have to be considered for computing the best candidate solution. In the annotation step (paragraph 3.2) information provided by the content of the input document and by the dependencies between two neighboring labels are used, and complex features, and especially features about the structure of the input document are used in the reranking step. This approach enables the use of a rich set of features while maintaining tractability. We will now detail features used in these two steps.

4.1 Features for the Annotation Step

CRFs let us specify features encoding local dependencies between the labels and arbitrary dependencies between labels and the input document. Features of interest for annotation include information about the leaf's content and its context in the input structure (like the number of siblings, the distance to the root node, ...).

Another useful information is provided by the data type specified by the schema. XML Schema [W3C, 2004],can specify, for each element, a type that describes the representation and the interpretation of values that can appear under this element. Knowing this information, we can restrict the potential labels of a given elements. Using this kind of feature is possible as the feature functions may depend on both the input observation sequence and on the output labels. Table 1 gives some typical examples of features used in our experiments.

4.2 Features for the Reranking Step

The reranking step ranks candidate solutions by considering global and long-distance dependencies. In our experiments, we used two main kinds of features.

The first one aims at taking into account statistics measured in the target documents of the training corpus: while the target schema describes which structures are *valid*, regularities observed in a set of documents following the target schema informs us about the real structures

content features	```begins-with-number```
	```begins-with-capitals```
	```contains-number```
	```contains-http```
	```contains-spaces```
	```contains-1-to-5-spaces```
	...

context features	```is-only-child```
	```has-1-to-3-siblings```
	```is-descendant-of-title```
	...

data type features	```is-xs_string```
	```is-xs_duration```
	```is-xs_time```
	...

Table 1: Example of features used to extract relevant elements from the sequence of leaves of the input document. Three kinds of features were used: features describing the content of a node (e.g. a node contains "http"), features describing the context of a node (e.g. this node has no siblings) and features describing the type of the data content in that node (e.g. this node content is a string according to the XML Schema specification).

observed in the collection. For instance, if in the target schema, a section may be composed of an arbitrary number of paragraphs, the observation that for most documents sections are made of 2 to 5 paragraphs is clearly useful. These regularities are measured using similarities found between structured candidate solutions for a new input document and the target outputs documents of the training set.

This similarity is computed by a *tree kernel*, k_{tree} [Collins and Duffy, 2001]. This kernel allows to take into account both local and long distance dependencies between document elements: the similarity between two documents is defined as the number of common tree fragments they share. The number of subtrees appearing in a tree grows exponentially with the size of the tree, but, thanks to a dynamic programming solution, detailed in[Collins and Duffy, 2001], this similarity can be computed without enumerating all subtrees.

We also used features encoding dependencies between input and output trees such as:

- a comparison between the number of nodes of the input document and the output document.

- the number of common *spans* between the input document and the candidate solution. The span of a node n is the pair (```starting point, ending point```) describing the part of the content sequence *covered* by the subtree whose root is n.

- the score of trees during the generation step.

The values of these global features are combined using a standard Radial Basis Function kernel denoted $k_{features}$.

The global kernel used in Equation 3 for the ranking step is then defined by a combination

of k_{tree} and $k_{features}$:

$$k\left((t, d^{\text{in}}), (t_{ij}, d_{ij}^{\text{in}})\right) = k_{tree}(t, t_{ij}) + \\ k_{features}\left((t, d^{\text{in}}), (t_{ij}, d_{ij}^{\text{in}})\right) \quad (4)$$

The addition of the two kernels expresses the concatenation of the two feature spaces.

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 [Doan and Halevy., 2005]. [Doan et al., 2003] describes one of the most complete approach which can handle both SQL and XML databases. In this work, the matching task is formulated as a supervised multi-label classification problem for scoring annotations. Traditionally, such matching has been approached mainly by finding pairwise-attribute pair. This approach works well for data-centric documents like the ones generally considered in the database community but cannot be readily generalized to deal with the highly-recursive Web documents whose structure is less regular. However, works on schema matching also stresses the need to consider several kinds of features (describing the content of documents, its structure, ...) in order to achieve good results.

In the Web community, the need for automatic HTML–XML transformation has been pointed out in many papers like [Breuel, 2003]. [Mukherjee et al., 2003] details how information conveyed by layout is closely related to the semantic structure of the document. This task shares some similarities with Wrapper Generation [Crescenzi et al., 2001], which attempts to recover, from training data, a set of regular expressions or a grammar and then use them for extracting desired information from Web pages. Wrappers do not handle content information and only consider syntactic features.

In the field of Natural Language Processing and of document annotation, various Machine Learning approaches have been proposed to extract relevant information from an input document. Most approaches are based on graphical models, like HMMs or CRFs [Lafferty et al., 2001], and can only exploit sequential dependencies among the labels.

Closer to our work, several approaches [Chidlovskii and Fuselier, 2005, Viola and Narasimhan, 2005, Shilman et al., 2005] have advocated the use of *probabilistic context-free parsing* to transform documents from a presentation-oriented to a *semantic-oriented* structure: they represent the document content as a sequence of observations and use a multi-class classifier and a Probabilistic Context Free Grammar to infer the tree structure of the output document. Several tree density models have also been proposed for different document structuring tasks [Gallinari et al., 2005, Denoyer et al., 2004]. While these approaches have achieved convincing results, they suffer several limits. First, they do not consider elements extraction: all content nodes that appear in the input document have to appear in the output document. Second, they can only consider local features, which makes impossible the use of information conveyed by the layout.

6 Experiments

Our model has been tested on two different corpora which are representative of two different application needs[2]. The first one is a collection of news published on LinuxFr, a popular technology-related Website/Internet forum. The data have been collected from Web pages and

---

[2]Instruction to generate the corpora can be found at http://connex.lip6.fr/ wisniewski/corpora

were converted to XML by hand. This corpus is a typical example of documents that can be found on news portals. Each article consists of a header describing metadata about the article (author, title, date, ...), a body containing the text and several threads of user-provided comments. The comments like the body of the article are structured and can use most of the HTML tags. The part describing users' comments is highly hierarchical and is a real challenge for structure extraction: the logical structure of the comments (which comment is the answer to which comment) has to be inferred from the layout structure of the input document.

The collection contains 2000 news with an average length of 392 content nodes and 281 inner nodes. News have, on average, 30 comments, and, at the most, one hundred comments. Only few elements were to be extracted: the author, the date, the title and the score. The main challenge of this task was to extract the logical structure of the comments.

The second corpus is based on the IMDb data. There are 4483 movie descriptions which have an average length of 70 content nodes and 90 inner nodes. The descriptions keep an attribute-value structure like in relational databases and there are only little textual data. As a result, this corpus has a fine-grained semantic but, at a the same time, a simple and regular structure. The target schema defines 7 complex elements (corresponding to relations) and 16 simple elements. The average size of target elements is 52 content elements.

6.1 Modus Operandi and Evaluation Measures

Each collection was randomly split in two equal parts for learning and testing. Experiments were performed on the two corpora using the same features (described in Section 4). The only parameter to be set is N, the number of candidate solutions generated. We conducted three series of experiments in which N was, respectively, 10, 50 and 100.

As baseline models we use a sequence labeling model (denoted "CRF" in Table 2) and the model of [Chidlovskii and Fuselier, 2005] (denoted "PCFG" in Table 2). The former corresponds to the CRF used in the first step of our method (Section 3.2) with the features described in Section 4.1. This model is only able to extract elements and not the relations between them. The model of [Chidlovskii and Fuselier, 2005] represents HTML documents as a sequence of observations corresponding to the HTML leaves of the document and uses a Maximum Entropy Classifier and a Probabilistic Context Free Grammar to infer the tree structure of the output document. This approach assumes that the content of documents is kept unchanged during the transformation (i.e.: the content nodes of the HTML document and of XML document are the same). In our experiment, we selected the best solution of the sequence labeling step described in Paragraph 3.2 that complies with the target schema as an input to the system. The features of the Maximum Entropy Classifier are the ones of the CRF; the non-local features described in Section 4.2 can not be used in this model.

For the evaluation of our results, the desired output document (t^{out}) is considered as the golden-standard. Then we measure the similarity between the desired and the computed outputs. As a similarity measure, we used the percentage of correctly determined constituents. A constituent is defined by a pair (`tag, span`) and describes both the label of a node and its position in the tree. For instance, the constituents of the subtree under the first `character` node of Figure 1 are: (`actor, 0, 1`), (`__DISMISS__, 1, 2`), (`role, 2, 3`) and (`character, 0, 3`). If the measure is equal to 100%, t^{out} has been found. This metric precludes errors to propagate: an element can be recognized even if one of its sub-element is not. We evaluate the performance on content nodes and inner nodes separately so as to distinguish errors made when identifying elements from those made when identifying relations between them. Because the distribution of labels for content nodes is unbalanced (the label indicating that a element must be ignored outnumbers other labels), we compute a micro and a macro

score for content nodes: the macro metric is the average of the results for each class while the micro metric weights the results of each class by its size. Here, macro is more significant.

6.2 Results

Table 2 presents the results of our experiments.

For leave nodes, the *simple* CRF model already achieves good performance, by considering only local features and local dependencies (in this case: between two adjacent labels). Taking into account the information provided by the schema clearly increases performance. This can be explained by the observation that CRFs solutions are biased toward the majority class for labels: using the constraints described by the schema allows this bias to be corrected. The proposed model offer a clear performance increase compared to the baseline solutions as can be seen from the macro scores on leaves. The labeling of leaves is almost perfect.

Results for inner nodes evaluate the capacity of our method to learn relations between elements. Performances show that using non local feature is important: the reranking approach outperforms the baseline model in both cases and provides an improvement of more than 40 points for the LinuxFr corpus. Note, however, that considering 100 candidate solutions, instead of 10 candidates, slightly reduces performances, presumably due to sparse data problems: ranking 100 elements is more complicated than ranking 10 elements. A closer analysis of the results shows that the system performs really well on the most regular part of the documents but performance falls on the hierarchical part of the corpus. In all cases, the reranking step depends on the quality of the candidate solutions.

	IMDb			LinuxFr		
	Leaves		Inner Nodes	Leaves		Inner Nodes
	macro	micro	macro	macro	micro	macro
CRF	78.5%	95.8%	—	80.2%	98.2%	—
PCFG	87.3%	98.1%	78.9%	90.1%	99.3%	24.2%
10-candidates	97.7%	99.3%	90.1%	98.1%	99.8%	67.3%
50-candidates	98.6%	99.4%	91.1%	99.7%	99.9%	74.6%
100-candidates	95.4%	98.9%	86.2%	99.3%	99.4%	64.1%

Table 2: Restructuring results for the two corpora. The score corresponds to the percentage of constituents that have been correctly recognized. Dashes indicate when the measure is not relevant.

7 Conclusion

We have proposed a general framework for the automatic conversion of loosely structured documents to a mediated XML schema. This framework allows us to use rich features characterizing local, global and long distance dependencies among the document elements. The proposed algorithm offers a good complexity compromise. It allows us to perform a large range of document structuration tasks at a reasonable cost. Prospective experiments were performed on two corpora and have shown promising results.

Several challenges are still to be taken up. First, the complexity of the generation step is still too high. We are currently investigating the use of approximate greedy search algorithms, like [Daumé III and Marcu, 2005], for reducing the time needed to generate solutions. A second major issue is to investigate how the information learned on one corpus can make learning easier on another corpus. In future work, we plan to extend our model to tackle these two problems.

8 Acknowledgments

References

David Aumueller. Semantic authoring and retrieval within a wiki. In *European Semantic Web Conference*, 2005.

Thomas M. Breuel. Information extraction from html documents by structural matching. In *WDA'03*, 2003.

Orkut Buyukkokten, Hector Garcia-Molina, and Andreas Paepcke. Seeing the whole in parts: Text summarization for web browsing on handheld devices. 2001.

Boris Chidlovskii and Jérôme Fuselier. Supervised learning for the legacy document conversion. In *DocEng '04: Proceedings of the 2004 ACM symposium on Document engineering*, pages 220–228, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-938-1. doi: http://doi.acm.org/10.1145/1030397.1030439.

Boris Chidlovskii and Jérôme Fuselier. A Probabilistic Learning Method for XML Annotation of Documents. In *IJCAI*, 2005.

M. Collins and N. Duffy. Convolution kernels for natural language. In *NIPS*, 2001.

Michael Collins and Terry Koo. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31:25–69, 2005.

Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of 27th International Conference on Very Large Data Bases*, pages 109–118, 2001. URL `citeseer.ist.psu.edu/ crescenzi01roadrunner.html`.

Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

Hal Daumé III and Daniel Marcu. Learning as search optimization: approximate large margin methods for structured prediction. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 169–176, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-180-5. doi: http://doi.acm.org/10.1145/1102351.1102373.

Ludovic Denoyer, Guillaume Wisniewski, and Patrick Gallinari. Document structure matching for heterogeneous corpora. In *Workshop SIGIR 2004*, Workshop on IR and XML, Sheffield, July 2004.

A. Doan and A. Halevy. Semantic integration research in the database community: A brief survey. *AI Magazine, Special Issue on Semantic Integration*, 2005.

Anhai Doan, Pedro Domingos, and Alon Halevy. Learning to match the schemas of data sources: A multistrategy approach. *Mach. Learn.*, 50(3):279–301, 2003. ISSN 0885-6125. doi: http://dx.doi.org/10.1023/A:1021765902788.

Patrick Gallinari, Guillaume Wisniewski, Maes, and Lud ovic Denoyer. Stochastic models for document restructuration. In *ECML Workshop on Relationnal Machine Learning*, 2005.

Yasuto Ishitani. Document transformation system from papers to xml data based on pivot xml document method. In *ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition*, page 250, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1960-1.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2000.

G. Kazai, N. Govert, M. Lalmas, and N. Fuhr. The INEX evaluation initiative. 2003.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001. URL `citeseer.ist.psu.edu/lafferty01conditional.html`.

Bing Liu and Kevin Chen-Chuan-Chang. Editorial: special issue on web content mining. *SIGKDD Explor. Newsl.*, 6(2):1–4, 2004. ISSN 1931-0145. doi: http://doi.acm.org/10.1145/1046456.1046457.

S. Mukherjee, G. Yang, and I. Ramakrishnan. Automatic annotation of content-rich html documents: Structural and semantic analysis. In *ISWC'03*, 2003.

Xuan-Hieu Phan and Le-Minh Nguyen. Flexcrfs: Flexible conditional random field toolkit, 2005. http://www.jaist.ac.jp/ hieuxuan/flexcrfs/flexcrfs.html.

Michael Shilman, Percy Liang, and Paul Viola. Learning non-generative grammatical models for document analysis. In *ICCV '05*, 2005.

Charles Sutton and Andrew McCallum. Collective segmentation and labeling of distant entities in information extraction. In *ICML workshop on Statistical Relational Learning*, 2004.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

Paul Viola and Mukund Narasimhan. Learning to extract information from semi-structured text using a discriminative context free grammar. In *SIGIR '05*, 2005.

M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer. Semantic wikipedia. In *In Proceedings of the 15th International Conference on World Wide Web (WWW'06)*, 2006.

Technical Comittee W3C. Xml schema recommendation, 2004. URL `http://www.w3.org/TR/xmlschema-0/`.

Matthew Young-Lai and Frank Wm. Tompa. Stochastic grammatical inference of text database structure. *Machine Learning*, 2000.