

# HTML-to-XML Migration by Means of Sequential Learning and Grammatical Inference

Boris Chidlovskii, Jérôme Fuselier

Xerox Research Centre Europe

6, chemin de Maupertuis, F-38240 Meylan, France

{chidlovskii,fuselier}@xrce.xerox.com

## Abstract

We consider the problem of document conversion from the layout-oriented HTML into a semantic-oriented XML annotation. An important fragment of the conversion problem can be reduced to the sequential learning framework, where source tree leaves are labeled with XML tags. We review sequential learning methods developed for the NLP applications, including the Naive Bayes and Maximum Entropy. Then we extend these methods with the Hidden Markov model (HMM) that injects the transition probabilities into the leaf classification function. Finally, we address the issue of HMM topology. We adopt grammatical inference methods to induce the HMM topology and show how to extend the sequential learning methods accordingly. We test all methods on a particular conversion case and report the evaluation results.

## 1 Introduction

The eXtended Markup Language (XML) has become a standard language for data exchange and reuse in various domains of data and content management, publishing and multimedia. Due to its extensible tag set, the XML addresses essentially the semantic-oriented annotation of the document content (titles, authors, references, tools, etc.), while all the rendering issues are delegated to reuse and re-purposing components which visualize the content, for example on different devices, with the help of appropriate XSLT scripts.

However, for companies and organizations that already own large legacy document collections, the migration toward XML represents a serious conversion problem. The legacy documents are often available in the electronic form, in one of the visualization-oriented formats like PDF, MS Word or (X)HTML that describe *how* to render the document content but carry little information on *what* the content is (catalogs, bills, manuals, etc.) and how it is organized.

The HTML-to-XML conversion conventionally assumes a rich target model given by an XML schema definition, in the form of a Document Type Definition (DTD) or a W3C XML Schema. The target schema describes the domain- or user-specific elements and attributes, as well as constraints on their usage, like the element nesting, an attribute uniqueness, etc.

The conversion of layout HTML annotations into the semantic XML may be achieved by structural transformations. However, because of the complexity of source documents and a frequent ambiguity of the layout annotation, the conversion task can be hardly automated without bringing in the process either the domain knowledge or an important set of examples that would instruct the human or a computer program how to produce the transformation rules.

<b>Domaines de compétences</b>
• Génie logiciel
<b>Formation</b>
• 2002 : <i>DEA Informatique</i> , Université de Savoie
• 2001 : <i>Maîtrise Informatique</i> , Université de Savoie

Figure 1: The example CV fragment.

In this paper, we adopt the supervised learning framework for the legacy conversion problem; so we assume that HTML documents are provided together with their target XML annotation. Each such pair (*source document*, *target document*) exemplifies the conversion process and forms an instance in the training set.

As an example, consider the conversion of Curriculum Vitae documents into the XML format. Figure 1 shows a fragment of a student CV that reports competence domains, studies, obtained degrees, etc. The content is presented in a way that eases the visual capture of information by a human. Figure 2.a shows the corresponding HTML source; it is a tree whose nodes are layout-oriented tags that instruct a browser how to render the document content. All tree leaves (given in bold) refer to content fragments given by PCDATA nodes, these fragments are called external or *content leaves*.

The result of conversion into the target XML is shown in Figure 2.b. The target tree provides the semantic annotation of the CV data. The tree leaves (given again in bold) refer to the content leaves “inherited” from the source tree.

## 2 Conversion model and sequential learning

Source HTML documents are essentially *semi-structured*; they enhance the free-form (unstructured) formatted text with

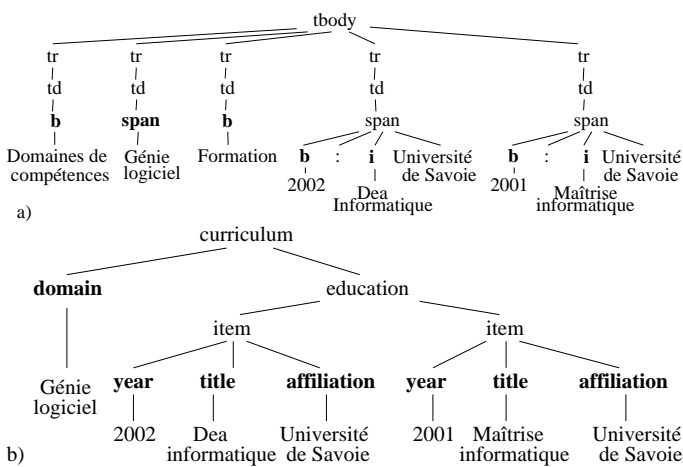


Figure 2: Tree representations of the source HTML (a) and target XML (b) fragments.

the meta information, given by HTML tags and attributes. As the document structure is essentially layout-oriented, the use of tags and attributes is not necessarily consistent with target elements. As an example, in Figure 2.a, text in bold (`<b>`) can be mapped into **year** target element or left off (leaves "Domaines de compétences" and "Formation"). The irregular use of tags in semi-structured documents makes the manual writing of document transformation rules difficult and error-prone.

In the supervised learning framework, the legacy conversion is learned from a training set of pairs of source HTML trees and the corresponding target XML trees. The tree correspondence is given by the *leaf alignment*, that maps each leaf in the target tree into the corresponding leaf in the source tree.

Although the learning models for the HTML-to-XML conversion remains our main goal [Chidlovskii and Fuselier, 2004], in this paper we are concerned with an important sub-problem, namely, tagging the leaves in source trees. Due to the leaf alignment of source and target trees, we introduce an *intermediate* class  $Y$  of sequences and address the problem of mapping the leaf sequence in a source tree into a sequence  $\mathbf{y} \in Y$ . The alphabet  $\Omega$  for  $Y$  is the set of all terminals in the target XML schema, extended with the class *None* for those source leaves that are not preserved in target trees. For the above example, we have  $\Omega = \{\text{domain, year, affiliation, title, None}\}$ .

In the following sections, we consider different learning models for labeling leaves sequences  $\mathbf{x} \in X$  of source documents with class sequences  $\mathbf{y} \in Y$ . The task can naturally be reduced to the sequential learning framework [Dietterich, 2002], where the training set  $S$  contains  $\{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \dots, n\}$ , where each item is a pair of (aligned) sequences  $(\mathbf{x}_i, \mathbf{y}_i)$ , where  $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{ik_i})$  are tree leaf sequences, and  $\mathbf{y}_i = (y_{i1}, y_{i2}, \dots, y_{ik_i})$  are class sequences.

In the CV example, one  $(\mathbf{x}_i, \mathbf{y}_i)$  pair consists of  $\mathbf{x}_i = (\text{"Domaine de compétence"}, \text{"Génie logiciel"}, \text{"Formation"}, \text{"2002"}, \dots, \text{"Université de Savoie"})$  and  $\mathbf{y}_i = (\text{None}, \text{do-}$

**main, None, year, . . . , affiliation**). Our goal is to construct a sequence classifier  $C$  that can correctly predict a new label sequence  $\mathbf{y}$  for a given input sequence  $\mathbf{x}$ ,  $\mathbf{y} = C(\mathbf{x})$ .

A number of sequential learning methods have been developed for the NLP and time-series applications, and our main interest is to investigate how these methods can be adapted to the leaf conversion task. By adopting the sequential principle, we attempt to estimate the conditional function  $P(y_t|y_{t-1}, x_t)$  that provides the probability of the current label  $y_t$  given the previous label  $y_{t-1}$  and current observation  $x_t$ . Under the conditional independence between the previous label  $y_{t-1}$  and the current observation  $x_t$ , we have

$$P(y_t|x_t, y_{t-1}) = P(y_t|x_t) \cdot P(y_t|y_{t-1}), \quad (1)$$

which is a joint evaluation of the leaf conditional function  $P(y_t|x_t)$  and the transition function  $P(y_t|y_{t-1})$  between two adjacent labels in the sequence  $\mathbf{y}$ .

In the following sections, we consider several groups of solutions for the function  $P(y_t|y_{t-1}, x_t)$ . We start by revisiting basic methods that evaluate the leaf conditional function  $P(y_t|x_t)$ . These methods ignore the sequential nature of  $\mathbf{x}$  and  $\mathbf{y}$  and consider them rather as sets; we will call these methods *stateless*. Then we consider the state transition function  $P(y_t|y_{t-1})$  based on the Hidden Markov model. Finally, we will discuss how to couple stateless methods with the HMM. We will describe in detail the MEMM method [McCallum *et al.*, 2000] which couples the Maximum Entropy principle with the HMM.

The HMM can learn the transition probabilities from the training set, but it assumes that the topology of the underlying automaton is given a priori. Existing methods, including the MEMM, use the basic approach, by which the automaton has the *one-state-one-class* topology, where each state is associated with one class  $y \in \Omega$ . To overcome the drawbacks of the one-state-one-class HMM topology, we extend further the MEMM method, by adding a component that infers the automaton topology from the label sequences  $\mathbf{y}$  in the annotation set  $S$ . For the topology induction, different methods from the grammatical inference can be used; in our experiments, we use the Alergia method [Carrasco and Oncina, 1994] for the identification of probabilistic state automata.

To characterize the methods presented in the paper, we follow [McCallum *et al.*, 2000] in using the corresponding dependency diagrams (see Figure 3). Coupling a stateless method  $X$  with the HMM is denoted as a  $X$ -MM method. Coupling the method  $X$  with the variable topology HMM is abbreviated as  $X$ -VMM. In particular, the combination of the variable topology HMM with the Maximum Entropy principle gives the MEVMM method.

### 3 Stateless methods

We start with methods that do not take into account the sequential nature of  $\mathbf{x}$  and  $\mathbf{y}$ . The dependency graph for a stateless method is given in Figure 3.a. Three stateless methods are presented in the following subsections.

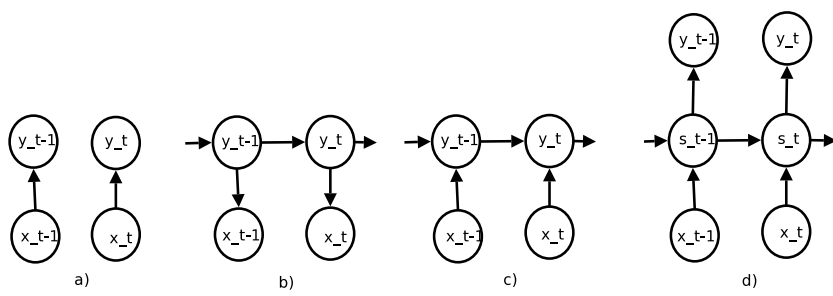


Figure 3: Dependency graphs. a) Stateless method X; b) HMM; c) X-MM methods; d) X-VMM methods.

### 3.1 Naive Bayes on content

We start with Naive Bayes classifiers which are widely used in traditional text classification systems, where the document content is only available for training. The model used for the representation of documents is the usual “*bag of words*” model; it simplifies the abstraction of the documents by only considering words frequencies, with an implicit criterion of independence between words. An observation (an HTML leaf)  $x$  in the sequence  $\mathbf{x}$  is represented by the frequencies of words contained in  $x$ . In the general case, we may consider lemmas instead of words; we may also suppress stop words which are usually irrelevant to a document.

Formally, a probabilistic classifier based on the Naive Bayes assumption tries to estimate the probability  $P(y|x)$ , the probability that the observation  $x$  belongs to the class  $y \in \Omega$ .

The *Maximum A Posteriori* estimator says that to achieve the highest classification accuracy,  $x$  should be assigned the class with the highest posterior probability :

$$y_{map} = \underset{y \in \Omega}{\operatorname{argmax}} P(y|x). \quad (2)$$

The Bayes theorem is used to split the estimation of  $P(y|x)$  into three parts:

$$P(y|x) = \frac{P(y) \cdot P(x|y)}{P(x)}. \quad (3)$$

As  $P(x)$  is independent of  $y$  values, it can be ignored and excluded it from the computation. The classification will then consist in resolving the following:

$$y_{bayes} = \underset{y \in \Omega}{\operatorname{argmax}} (P(y) \cdot P(x|y)). \quad (4)$$

The likelihood  $P(x|y)$  may be simplified due to the word independence assumption. An observation  $x$  is composed of a words  $w$ , as we can rewrite the equation (4) :

$$y_{bayes} = \underset{y \in \Omega}{\operatorname{argmax}} (P(y) \cdot \prod_{w \in x} P(w|y)). \quad (5)$$

In (5), the prior  $P(y)$  and the likelihoods  $P(w|y)$  are both easily calculated by counting the frequencies in the training set. The prior does represent the distribution of the classes in the training set, and the likelihood gives the probability of a word given a fixed class.

### 3.2 Naive Bayes on HTML structure

Another way to apply the Naive Bayes approach is to build a classifier using the *structure* of the source HTML instead of the leaf content. The idea is to capture the structural information surrounding a leaf in the tree. One way to achieve this is to represent a leaf as a “*bag of paths*”, instead of “*bag of words*”. For this, we use the notion of reverse paths on tree leaves. A *reverse path* is a special case of a simple path in a tree; it defines a path of non-zero length which starts at a given tree leaf and terminates either in an internal node or a different leaf. For a given leaf  $x$  in a source tree, we consider the set  $RevP(x, l)$  of all reverse paths of a given length  $l$ . In a tree, any pair of nodes has a unique simple path, so all paths in  $RevP(x, l)$  start at leaf  $x$  but terminate in different nodes. The set of reverse paths is finite and reflects the structural context of  $x$ .

Any reverse path in  $RevP(x, l)$  set can be seen to a word in the alphabet of tag names. This raises the problem of encoding the reverse paths in unranked trees. Due to the undefined number of sons, it is insufficient to encode the paths using the tag names only. For instance, in a subtree  $t = A(BCB)$ , we may want that the path between a node  $A$  and the first son  $B$  is encoded differently from the path between  $A$  and the third son  $B$ . We solve this problem by rewriting the unranked source trees into binary ranked trees, using the binarization technique from [Neven, 2002]. In a binary tree, we can explicitly use the left, right and upward edges for encoding reversed paths.

An additional source of structural information is the values of *attributes* in the tree nodes. In layout-oriented documents, node attributes and their values can describe a valuable and rich information relevant to a leaf and its context. By considering the trees from the XML Document Object Model (DOM) point of view, we define attributes as nodes of the source trees. This allows us to extend the technique of reverse paths to take into account the attributes and their values in the same manner as we did with the tree nodes.

### 3.3 Maximum Entropy

According to the Maximum Entropy principle, the best model to estimate probability distributions from data is the one that is consistent with certain constraints derived from the training data, but otherwise it makes the fewest possible assumptions. In the probabilistic framework, the distribution with the “fewest possible assumptions” is the one with the highest

Feature group	Features
Content features	length, number_of_numeric_words, number_of_words, number_of_uppercase_chars, number_of_digits, number_of_lowercase_words, number_of_char_words, number_of_char_numeric_words, etc.
Structural features	father_tag_name, grandfather_tag_name, next_sibling_father_tag_name, etc.
Attribute features	all attributes like font, width, font-color, height, x, y, emphasis, etc., for certain nodes surrounding a leaf like father, grandfather, etc.

Table 1: Three types of features for the ME classifier.

entropy, and closest to the uniform distribution. Each constraint expresses some characteristics of the training data that should also be present in the learned distribution [Berger *et al.*, 1996]. The constraint is based on a binary feature, it constrains the expected value of the feature in the model to be equal to its expected value in the training data.

One important advantage of Maximum Entropy models is their flexibility, as they allow to combine any set of syntactic, semantic and pragmatic features. Each feature  $f$  is binary and can depend on  $y \in \Omega$  and on any properties of the input observation  $x$  in  $\mathbf{x}$ . In the case of conversion, we define three sets of features:

1. *Content features* that express properties on text in leaves, like:

$$f_1(x, y) = \begin{cases} 1 & \text{if } y=\mathbf{year} \text{ and } x \text{ has only numeric} \\ & \text{characters,} \\ 0 & \text{otherwise.} \end{cases}$$

2. *Structural features* that capture the tree context surrounding a leaf, like:

$$f_2(x, y) = \begin{cases} 1 & \text{if } y=\mathbf{affiliation} \text{ and } x\text{'s father tag} \\ & \text{is span,} \\ 0 & \text{otherwise.} \end{cases}$$

3. *Attributes features* that capture the values of node's attributes in the source tree, like:

$$f_3(x, y) = \begin{cases} 1 & \text{if } y=\mathbf{domain} \text{ and the value of the font} \\ & \text{attribute of } x\text{'s father is times,} \\ 0 & \text{otherwise} \end{cases}$$

For the Maximum Entropy (ME) classifier, we define and extract about 90 raw features, some of them are presented in Table 1. Not all of these features may exist or be relevant in a specific collection. A feature is removed from the model if it presents no different values in the training set.

With the constraints based on the selected features  $f_j(x, y)$ , the Maximum Entropy method attempts to max-

imize the conditional likelihood of  $P(y|x)$  which is represented as a log-linear model:

$$P(y|x) = \frac{1}{Z(x)} \exp \left( \sum_j \lambda_j \cdot f_j(x, y) \right), \quad (6)$$

where  $Z(x)$  is a normalizing factor to ensure that all the probabilities sum to 1:

$$Z(x) = \sum_y \exp \left( \sum_j \lambda_j \cdot f_j(x, y) \right). \quad (7)$$

In experiments, we use the L-BFGS (Quasi-Newton) algorithm to evaluate the parameters  $\lambda_j$  of the ME models [Malouf, 2002].

## 4 Sequential methods

The second component of the probabilistic sequential function in (1) should evaluate the transition probability function  $P(y_t|y_{t-1})$  between adjacent labels in  $\mathbf{y}$ . For this goal, we use the Hidden Markov Model which is a generative model for  $\mathbf{x}$  and  $\mathbf{y}$  sequences.

To estimate the transition distributions  $P(y_t|y_{t-1})$ , the HMM observes all pairs of adjacent  $y$  labels in the training set and uses the Baum-Welsh algorithm which is a special case of the iterative Expectation Maximization algorithm.

The HMM can be applied to compute the probability of any particular  $\mathbf{y}$  for a given sequence  $\mathbf{x}$ . Moreover, the HMM can determine  $\mathbf{y}$  with the highest probability. This is achieved by the Viterbi dynamic programming algorithm [Rabiner, 1986] which computes, for each time step  $t$ , the probability of the most likely path  $(y_0, \dots, y_t)$  in the automaton starting at time step 0 and ending at time step  $t$ . When the algorithm reaches the end of the sequence  $\mathbf{x}$ , it returns the most likely sequence  $\mathbf{y}$  explaining how  $\mathbf{x}$  is generated (see the dependency graph in Figure 3.b).

HMMs provide an elegant and sound methodology for the sequential learning, but they suffer from one principal drawback: the HMM is often a poor model of the process producing the data. The problem stems from the Markov independence property that makes difficult to capture any relationship between two non-adjacent  $y$  values [Dietterich, 2002]. To overcome the limitations of the HMM, several directions have been explored. Some, like Maximum Entropy Markov models [McCallum *et al.*, 2000], do not try to explain how the  $\mathbf{x}$  data is generated; they instead try to predict  $\mathbf{y}$  given  $\mathbf{x}$ . This permits them to use arbitrary features of the observations  $x$  including features describing non-local interactions.

### 4.1 Coupling stateless methods with HMM

The Maximum Entropy Markov models (MEMMs) couples the Maximum Entropy stateless method with the HMM; the corresponding dependency graph is given in Figure 3.c. The MEMM merges the HMM transitions and content observations in a single conditional function  $P(y_t|y_{t-1}, x_t)$  that provides the probability of the current label  $y_t$  given the previous label  $y_{t-1}$  and observation  $x_t$ . In the MEMM, features

do not depend only on the observation but also on the label predicted by the function being modeled. Thus, each MEMM feature  $f^*$  is a function of three arguments, an observation  $x_t$ , a possible next label  $y_t$  and the previous label  $y_{t-1}$ . If  $f_1$  is a ME binary feature defined in the previous section, then the MEMM feature  $f_1^*$  may be defined as follows:

$$f_1^*(x_t, y_t, y_{t-1}) = \begin{cases} 1 & \text{if } y_1(x_t, y_t) \text{ and } y_{t-1} = \text{domain} \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

The MEMM learns the conditional function  $P(y_t|y_{t-1}, x_t)$  by applying the Maximum Entropy principle that attempts to maximize the conditional likelihood of the data,  $P(\mathbf{y}|\mathbf{x})$ . The Maximum Entropy approach allows to represent the conditional function as a log-linear model:

$$P(y_t|y_{t-1}, x_t) = \frac{\exp(\sum_j \lambda_j \cdot f_j(x_t, y_t, y_{t-1}))}{Z(y_{t-1}, x_t)}, \quad (9)$$

where  $Z(y_{t-1}, x_t)$  is a normalizing factor to ensure that the probabilities sum to 1.

As the next label  $y_t$  depends on both observation  $x_t$  and the previous class  $y_{t-1}$ , the MEMM in [McCallum *et al.*, 2000] uses the HMM with the basic topology, where one state corresponds to one class  $y \in Y$ , and associates one Maximum Entropy classifier with each state. The MEMM training consists in 1) grouping all observations  $x$  that share the same previous class  $y$  and 2) training a ME model with all the observations in the group. In total, the MEMM contains as many ME models as the number of classes in  $\Omega$  and one ME model for the start state.

Although the MEMM shows good results in various information extraction tasks, in some cases, its performance is similar to or even worse than the performance of the stateless ME method. The problem is partially linked to the inappropriate topology of the automaton. The basic topology can penalize the overall MEMM performance and even erase all advantages offered by the conditional transition model.

The MEMM particularly suffers in the presence of many classes in  $Y$ . In this case, the partition of the training data into multiple groups reduces the amount of observation data associated with any automaton state and therefore considerably increases the risk of inaccurate parameter estimation.

In the following section we investigate the influence of the automaton topology on the overall performance of coupled methods. We replace the one-state-one-class approach with a more flexible one where the automaton topology is not given a priori, but induced from the annotated data.

## 5 Topology inference

We refine the MEMM method by allowing the learning system to reason about the topology of the finite-state probabilistic automaton. Going beyond the basic one-state-one-class case, the system can associate, on one side, several states with one class  $y$ ; on the other side, several classes can be regrouped together in one automaton state.

We allow the system to infer the automaton topology from the annotated corpus. We assume that the class sequences  $\mathbf{y}_i$

in the training set represent the positive examples of a probabilistic regular language. Then we can use these sequences to find the minimum deterministic finite automaton which generates the language.

We adapt one of the standard methods for inferring the probabilistic automata from positive examples, namely Alergia [Carrasco and Oncina, 1994]. The Alergia method first builds the most specific automaton that accepts the  $\mathbf{y}_i$  sequences. Then the algorithm generalizes the automaton by state merges. The state merge routine uses a theoretically grounded criterion for the state equivalence based on observed termination and transition frequencies of a pair of states. Any merge of two states recalculates all the frequencies and can trigger new merges. The recursive routine runs until no merge is possible. Once the automaton topology is returned by the Alergia, the HMM algorithms retrain the transition and emission probabilities from scratch.

The X-VMM method that couples a stateless method X with the variable topology HMM has a dependency graph presented in Figure 3.d. The graph extends the X-MM graph by introducing an intermediate level of (hidden) states  $s_t$ . The probabilistic sequential function for the X-VMM method is defined as follows:

$$P(y_t|s_{t-1}, x_t) = P(s_t|s_{t-1}, x_t) \cdot P(y_t|s_t), \quad (10)$$

which is a direct generalization of (1). Indeed, the X-MM graph (Figure 3.c) is a special case of the X-VMM graph (Figure 3.d) where any state  $s_t$  is associated with class  $y_t$ .

By extending the method toward the variable topology, we have implemented three variations of the MEVMM method:

1. *The Random Approach* : To verify how the variation of the automata topology can influence the performance of a method, we first allow random groups of classes to form the HMM states. The method is simple, we define the number of states in the automaton and let the system to randomly produce an automaton with the given number of states and with a list of classes associated with the states.
2. *The Cluster Approach* : The basic automaton topology in MEMM divides the training instances according to their previous. Due to the skewed class distribution in the collection, some classes have so small training sets that we are not able to train a good ME classifier. The heuristics consists in grouping together the small classes in the basic MEMM whose training sets are under a certain threshold.
3. *The Inference Approach* : This version generates the automaton topology by using the Alergia algorithm presented above.

## 6 Evaluation

We have run series of experiments on classifying HTML leaves on a collection named TechDoc; it includes 60 technical documents from car repairing manuals, available in both HTML and XML formats. Target documents have a fine-grained semantic granularity; the target schema is given by

one complex DTD that includes 27 terminals and 35 nonterminals. The collection is quite heterogeneous and presents multiple inconsistencies between any two different documents.

We test three groups of methods presented in Sections 3-5. The first group include three stateless classifiers from Section 3, *i.e.* the Naive Bayes on content, Naive Bayes on structure and Maximum Entropy.

The second group includes two versions of the structural classifier. Both versions evaluate the class transition probabilities, they are trained with class sequences  $\mathbf{y}$  from the training set. The first version is the one-state-one-class HMM, with 29 states (27 states for all classes, one start and one end states). The second version is a probabilistic automaton resulting from the Alergia algorithm with the confidence threshold for the state equivalence set to 0.95. In the cross validation mode with the folding 4, the automaton contains 14 to 16 states.

The third group covers all methods of coupling the Maximum Entropy stateless method with different structural methods. We first test the MEMM classifier which is a combination of the ME classifier with the one-state-one-class HMM. Using the MEVMM-Random method, we have created 30 different HMM topologies; we report the performance of the best and the worst cases among them. The MEVMM-Cluster method builds the automaton by grouping small size states in the MEMM. The method deploys a simple heuristics for finding out cluster-like groups of small size states (less than 3% of the data set); this allows us to reduce the number of states from 29 to 14. Finally, the MEVMM-Inference method is a combination of Maximum Entropy classifier with the automaton produced by the Alergia method.

## 6.1 Evaluation metrics

In the conversion problem, we often deal with the leaf classification problem on a large number of classes. For each external leaf in a source tree, we estimate the most appropriate class using different methods, and we need a measure to determine how well each of these methods classify the source leaves. Traditional precision/recall measures that determine the ratio of correctly classified instances and the ratio of classified instances are well suited for the binary classifiers. However, they are not well appropriated for the multi-class classification, particularly, like in our case, when certain classes cover less than 0.1% of source leaves.

We thus adopt the *cost-based classification model* [Zaki and Aggarwal, 2003] to evaluate the leaf classification methods. The accuracy of a classification model  $M$  on a data set  $D$  is given by the ratio of correct predictions to the total number of predictions made :  $\alpha(M, D) = \frac{\eta(D)}{|D|}$ . To deal with multi-class problems, for each class  $\omega_i$ , we link a weight  $w_i$  with the constraint that  $\sum_{i=1}^k w_i = 1$ . The cost-sensitive accuracy, denoted  $\alpha^{cs}$ , is defined as the weighted average of the accuracy of the classifier on each class. Formally, we define  $\alpha^{cs} = \sum_{i=1}^k (w_i * \alpha(M, D_i))$ .

There are several cost-models that one could use to com-

Type	Method	TechDoc	
		$\alpha_{prop}$	$\alpha_{eq}$
Stateless	Naive Bayes on content	73%	64%
	Naive Bayes on structure	82%	79%
	Maximum Entropy	87%	83%
Structure	HMM - 1-state-1-class	28%	8%
	Variable topology HMM	31%	10%
Coupling	MEMM	85%	83%
	MEVMM-Random min	84%	82%
	MEVMM-Random max	<b>91%</b>	<b>90%</b>
	MEVMM-Cluster	85%	84%
	MEVMM-Inference (Alergia)	90%	87%

Table 2: Leaf classification for the TechDoc collection.

pute the classification accuracy. We use two of them as they appear to better capture the method behavior on the data:

- The *proportional model* uses  $w_i = \frac{|D_i|}{|D|}$ , *i.e.*, weights are proportional to the probability of the class in  $D$ . This is the nearest model from the precision measure.
- The *equal model* uses  $w_i = \frac{1}{k}$ , *i.e.*, all classes are weighted equally.

## 6.2 Evaluation results and discussion

Table 2 reports the results for all methods on the TechDoc collection. Among the stateless methods, the Maximum Entropy is the best classifier as it offers a robust model to integrate different types of features and to take advantage of all the information we extract. Although the Naive Bayes approaches impose a strong independence hypothesis between word lemmas or reverse paths, we see that both NB classifiers perform quite well. The major concern is that they work on two different and incompatible feature sets.

As we expected, the purely structural methods perform poorly as compared to the content-based methods. The explanation lies in the fact the methods predict classes only from the transition probabilities and disregard all the information in the source documents.

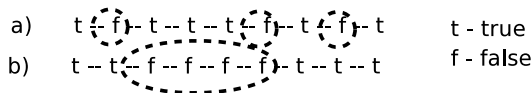


Figure 4: Typical misclassification patterns; a) ME; b) MEMM.

As Table 2 shows, coupling the content-based methods with structural methods does not always improve the accuracy. Overall, the MEMM performance is not on the level of our expectations; it is even less accurate than the ME alone. The analysis of errors committed by the MEMM indicates that the problem is linked to the skewed class distribution. Very small training sets for some classes leads to inaccurate transition probabilities in the automaton. As result, once the

classifier makes an error, it may follow the wrong transition sequence through the automaton during multiple steps, before it happens to return on the right path. Figure 4 shows how the patterns in the sequence classification with the MEMM is different from the ME. In the case when the transition probabilities are greater than the ME probabilities, the MEMM classifier makes more errors than the ME alone. As conclusion, the basic one-state-one-class topology does not work well in the presence of multiple classes and the skewed class distribution.

The MEVMM-Random version is interesting because it helps estimate if certain automaton topologies can improve the ME classifier in a significant way. Moreover, it suggests how the variable topology can influence the MEMM though it can not guarantee a good topology for a specific collection. The MEVMM-Inference method tries to overcome this problem by generating an automaton consistent with the training set, more precisely which is consistent with the class sequences. By observing the results, we can see that it is a good compromise between the original MEMM and the random one. It is not as good as the best random topology but it is still better than both ME and MEMM classifiers. We can conclude that the grammatical inference may help us to create a good automaton topology although it can be further improved.

## 7 Conclusion

We have addressed the problem of the HTML-to-XML conversion. Under the order preserving assumption, we have reduced the problem to the problem that allows us to reduce the classification of the source tree leaves to the problem of the sequential learning. We have reviewed a number of learning methods, including ones from the sequential learning, and how they can be used for the leaf sequence annotation. We have presented preliminary results of our experiments with different methods on a real document collection, The analysis of the results show Although the last approach is not optimal, it is a good start for the improvement and allows us to define a reasonable operating mode to infer an efficient topology for the MEVMM classifier.

## References

- [Berger *et al.*, 1996] Adam L. Berger, Stephen Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.
- [Carrasco and Oncina, 1994] Rafael C. Carrasco and Jose Oncina. Learning stochastic regular grammars by means of a state merging method. In *International Conference on Grammatical Inference*, pages 999–999. Springer-Verlag, September 1994.
- [Chidlovskii and Fuselier, 2004] Boris Chidlovskii and Jérôme Fuselier. Supervised learning for the legacy document conversion. In *ACM Symposium on Document Engineering, October 2004, Milwaukee, WI*, pages 220–228, 2004.

- [Dietterich, 2002] T. G. Dietterich. Machine learning for sequential data: A review. In T. Caelli, editor, *Lecture Notes in Computer Science*. Springer-Verlag, 2002.
- [Malouf, 2002] R. Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proc. 6th Conf. on Natural Language Learning*, pages 49–55, 2002.
- [McCallum *et al.*, 2000] Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proc. 17th International Conf. on Machine Learning*, pages 591–598. Morgan Kaufmann, San Francisco, CA, 2000.
- [Neven, 2002] Frank Neven. Automata Theory for XML Researchers. *SIGMOD Record*, 31(3):39–46, 2002.
- [Rabiner, 1986] Juang B. Rabiner, L. An introduction to hidden markov models. *IEEE Acoustics, Speech and Signal Processing Magazine*, 3:4–16, 1986.
- [Zaki and Aggarwal, 2003] M.J. Zaki and C. Aggarwal. XRULES : An effective structural classifier for XML data. In *9th International Conference on Knowledge Discovery and Data Mining, Washington, DC*, 2003.