Pattern Anal Applic (2004) 7: 66–76
DOI 10.1007/s10044-004-0208-3

ORIGINAL ARTICLE

Gy. Hévízi · T. Marcinkovics · A. Lőrincz

Improving recognition accuracy on structured documents by learning structural patterns

Received: 23 October 2002 / Accepted: 10 February 2004 / Published online: 30 March 2004

Abstract In this paper, we present a probabilistic method that can improve the efficiency of document classification when applied to structured documents. The analysis of the structure of a document is the starting point of document classification. Our method is designed to augment other classification schemes and complement pre-filtering information extraction procedures to reduce uncertainties. To this end, a probabilistic distribution on the structure of XML documents is introduced. We show how to parameterise existing learning methods to describe the structure distribution efficiently. The learned distribution is then used to predict the classes of unseen documents. Novelty detection making use of the structure-based distribution function is also discussed. Demonstration on model documents and on Internet XML documents are presented.

Keywords Bayesian networks · Classification · Novelty detection · Probabilistic tree model · XML

Introduction

XML has become one of the major standard components of today's Internet. It provides the functionality to mark up documents with arbitrary meaningful tags and model hierarchical document structures. XML makes additional structural and semantic information available on the Web in addition to text content. We think that XML, as a building block for the Semantic Web [1], will still be present everywhere for the foreseeable future.

Besides XML, over 80% of the data on the Internet is generated from relational databases [2], which have a well defined structure. Thus, the structural information of a web document is often specific to the source from where it originated.

The structural information present on the Web could be used to improve document retrieval results for search engines, or to enable more sophisticated content classification for portals and other integrated systems (e.g. Yahoo and Knowledge Portals). Most of the previous research on information extraction from HTML or XML documents used grammatical inference techniques [3, 4], specifying rules or regular grammars for XML documents. In contrast to these, we have examined a method based on probabilistic inference.

In this paper, we focus on designing a machine learning method that can extract probabilistic information about the structure of XML documents. Our learning system could be used as both a component of a complex classifier system that makes use of semantic information using several XML standards or as a part of simple wrapper programs [5]. Other methods dealing with the content of the XML and its semantic aspects based on existing technologies are briefly described in Sect. 6. An envisioned complex validation tool is sketched in Sect. 2, which incorporates our probabilistic structural pattern recognition module into existing validation technologies. In this envisioned architecture, every module has the right to make a decision if it can, otherwise, it passes the document on to the next module. A document may stay 'ambiguous' after being processed by every module.

The motivation of our approach is that traditional validation tools filter out information that has even the slightest error. Such approaches cannot be used if data could be important but are not encoded properly. Also, distributed storages may have partially corrupted data; either temporal or fatal breakdowns, as well as malicious attacks can give rise to problems. Then, restoration of the data is an important issue. The first step of restoration is a traditional validation to select the uncorrupted data, which should be followed by a probabilistic classifier that makes estimations. Such estimations may include the probability that the data can be restored or

G. Hévízi · T. Marcinkovics · A. Lőrincz (✉)
Department of Information Systems,
Eötvös University, Pázmány Péter sétány 1/C,
1117 Budapest, Hungary
E-mail: lorincz@inf.elte.hu

not. Also, the likelihood of belonging to different classes can decrease the computational time of restoration considerably. Last but not least, probability estimations having been trained on large data sets can provide estimations such as: this is not corrupted data, this data is not likely to belong to any of our classes, this data is novel to the database that has been used for training, etc. Such recognition of novelty can be of importance for monitoring intrusions for example. The probabilistic tool proposed here can be used in conjunction with other data restoration and data integrity monitoring techniques, and is suited to XML structures.

Preliminaries

Integration of a structure-based method and other technologies

There is already a wealth of XML technologies that can be used for classification. Here, we provide an overview of currently used syntactic and semantic tools that form the context of our method. A possible scenario for an XML classifier could be the following: a pre-processor system based on wrappers and other utilities (e.g. HTML Tidy http://www.w3.org/People/Raggett/tidy/) provides an input XML document.

Schema processors

Standard XML tools, validating parsers or schema processors could verify syntactic and grammatical constraints. A schema is a grammar which describes the structure and tag names in the document, along with other constraints (such as range checking of element values, etc.). There are several widely used XML schema languages [6]. If an XML document has a schema associated with it, and it conforms to the schema, it is considered valid and is classified into the category defined by the schema. One can even match an XML document to a set of schemas, as done in [7]. This scenario can arise when an XML document conforms to multiple standards, such as cXML or xCBL (http://www.xcbl.org), for example, in an e-commerce application.

Semantic component

Research on the Semantic Web has developed ontologies to provide shared knowledge and conceptualisation of data. Ontologies provide common vocabularies for concepts and describe general relationships between them to enable systems to infer additional knowledge which is not present in a document. Based on an ontology such as WordNet or EuroWordNet [8], inconsistency of tag names could be handled by a semantic component of a classifier. A real-world, ontology-driven indexing method using this technique is described by [9].

There are, however, still some cases where these approaches fail. Ontologies are language-specific and do yet not cover enough domain information. There is a clear trend to publishing online knowledge bases, such as openCYC (http://www.opencyc.org). It may be that tag names cannot be mapped into an ontology or that tag names cannot be resolved by a thesaurus (such as 'mgrno', an abbreviation for 'manager number' in an XML document generated from a database by using the name of a table column). The problem with schema-based approaches is that schemas try to enforce a strict grammar. Assume that a set of document type definitions (DTDs) were generated to classify an XML document class. These would fail to recognise the similarity of an XML document originating from a different source and having an additional element. Clearly, there can be cases when a category cannot be represented by a grammar or by a set of grammars. We list a few possible cases when rule-based methods can fail:

- The database (e.g. the worldwide web) undergoes continuous updating and extension, whereas the modification of the XML parser is delayed
- The database has been damaged or deleted and information is missing
- A set of different databases is to be transformed into a common new structure

Complex architectures

These are examples of technologies that have proved to be useful in the processing of XML data. They can be integrated into complex architectures in several ways. Figure 1 shows an example of this. The complex system performs validation. Each module has a set of models or strict rules that describe document types and these models can determine either acceptable documents or document types that should definitely be rejected. When a document is recognised with a high probability by one of these rules, a decision can be made. If none of these rules are satisfied by the document, or if the discrepancies are above the threshold, the module cannot take the responsibility for making a decision. It can also happen that different rules can label the very same document as 'acceptable' and 'to be rejected'. In the latter cases, the document can be marked 'ambiguous' by the module and can be passed to the next module.

Theoretical considerations

Probability distribution model for XML documents

Our objective is to exploit structure-based information from XML documents and provide methods using this information to facilitate classification and novelty detection on XML document sets. The first decision to be made is how we transform documents into a vector

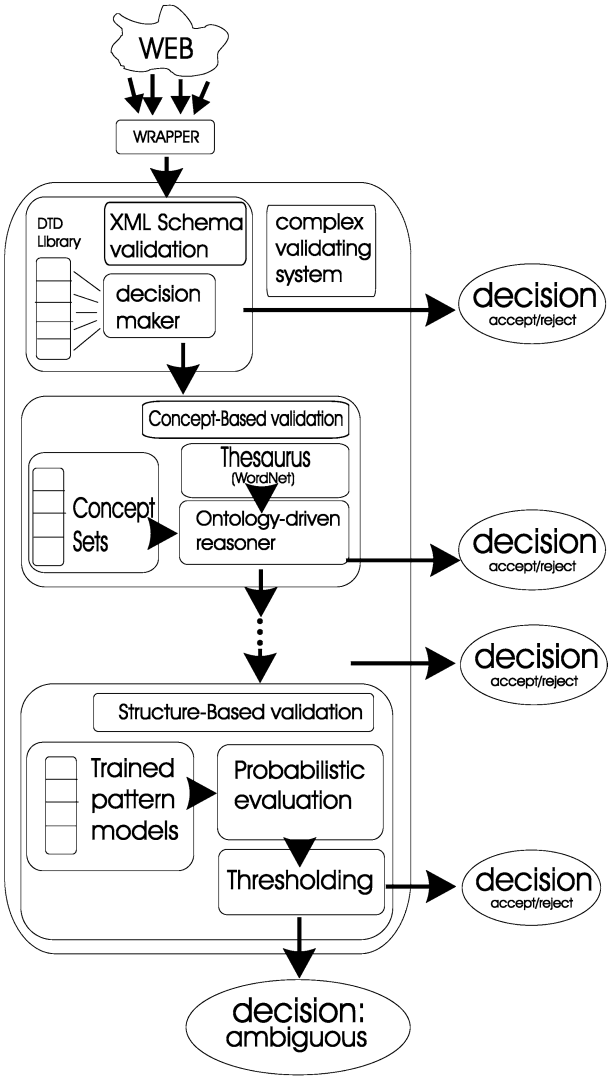

Fig. 1 An example of a possible complex XML processing architecture. One syntactic and one semantic method is placed in the architecture. An arbitrary number of other methods can be inserted into the sequence. Our structural pattern recogniser is placed at the end of the procedures. This type of sequential processing of documents ensures that every aspect is examined before the document is labelled 'ambiguous' or 'unambiguous', but the decision is made earlier whenever possible. By optimising the order of the modules, minimal CPU time consumption can also be ensured

space. In this paper, we discuss a method that relies on the structure and ignores the other properties of the document. We shall assume that the document has a tree structure, encoded in XML. A particular document structure will be mapped to a configuration of stochastic variables related to the tree structure. The result will be called the 'probabilistic tree model'.

An example for a tree-structured XML document is shown in Fig. 2a. There are different depths and different branching ratios along different paths. Such variations give rise to a dilemma: assume that we have an upper bound on the branching ratios, but the typical branching ratio is usually much smaller. To represent all possible existing nodes requires a very high-dimensional vector space, or a great number of probabilistic variables. We shall have a very large probability table with only a few non-zero entries. Tree classification is based on probability distributions and, from this point of view, such *raw* representation has two drawbacks:

1. The representation is large
2. The large number of poor probability estimations may spoil classification

A more compact (less detailed) representation of the tree structure can be used here. Such simplification can be justified by both practical and by theoretical arguments:

1. Probability estimations can be derived from the specific database. These estimations can be used to determine the optimal level of simplification
2. Computation time of the classes of documents may be restricted
3. Independent sets of variables may exist and can be found by novel methods of independent component analysis [10, 11, 12, 13, 14].

Such independent component analysis can break a tree into sub-trees. Two (out of many) possible simplifications are described and studied here.

Our first simplifying procedure is as follows. We fix the number of tags represented to a predefined number, say m. If more than m tags are present, then only the first m tags are kept. If less than m tags are present then tags with 0 values are introduced. This representation provides information about the presence or absence of tags. This method prescribes the structure of the tree and represents the stochastic property of the structure via the probabilistic variables at the vertices of the tree. The variables, $X^n=(X_1,...,X_n)$, are indexed according to breadth-first traversal, subject to the constraints introduced before.

Definition 2.1

$X_i=0$ There is no tag in this position

$X_i=1$ There is a tag in this position

The graphical representation will be called the basic constrained structure (BCS). The procedure will be referred to as the limiting-augmenting binary (LAB) representation on the BCS (e.g. the LAB representation of an XML document). The LAB representation for branching ratio = three is shown in Fig. 2b.

A more sophisticated approach can take the number of attributes into account. In turn, a distribution on attribute numbers can be estimated. This approach requires more CPU time, but has an increased representational capability. Probabilistic variables of this limiting-augmenting integer (LAI) representation on the BCS can be defined as follows:

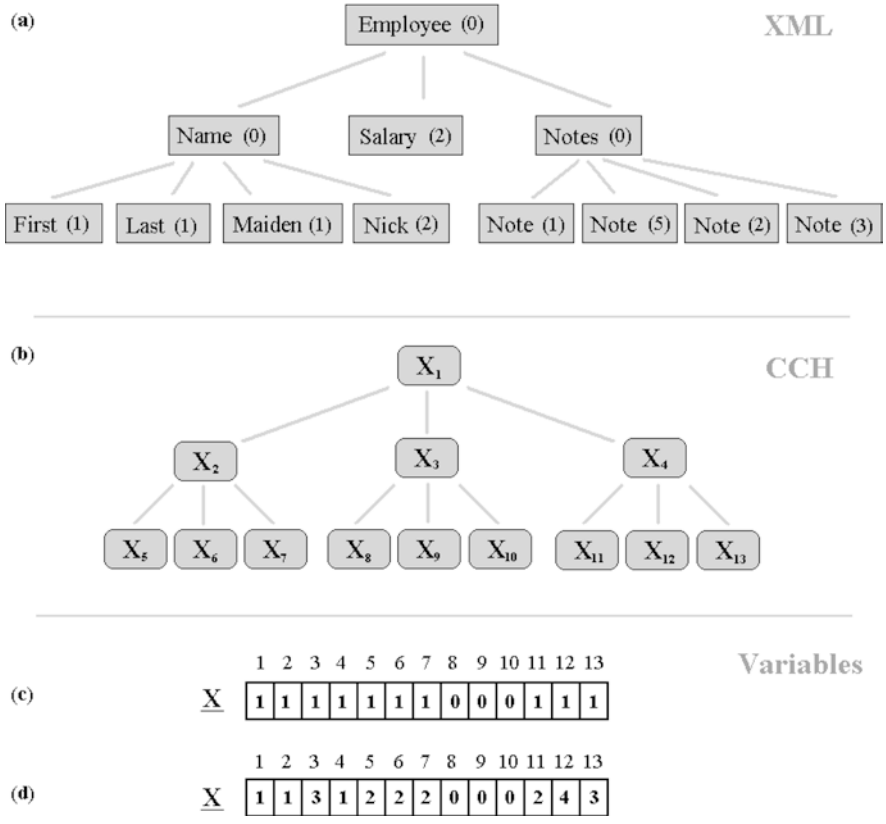

Fig. 2a–d Modelling document structures. **a** An example of a structured XML document. The numbers in brackets show the number of the tag's attributes. **b** A basic constrained structure (BCS) tree with depth = 2 and branching ratio = 3. **c** The limiting-augmenting binary (LAB) representation (see Definition 2.1). **d** The same representation with integer variables (LAI representation) with $k = 4$ (see Definition 2.2)

Definition 2.2

$X_i = 0$	There is no tag in this position
$X_i = 1$	There is a tag, but it has no attributes
$X_i = 2$	The tag has 1 attribute
...	...
$X_i = k$	The tag has $(k-1)$ or more attributes

Obviously, Definition 2.1 is a special case of Definition 2.2 with $k = 1$. Figure 2c, d depicts the structure coding of the LAB and the LAI methods on a sample XML structure, respectively. The figure also demonstrates the parameter-dependence of representational capability and resource requirements. Note that, with the choice of branching factor = 3, 'Nick' and the last 'Note' tags are not represented. This is a compromise between representational power and the constraints of our estimation. In the LAI representation on the BCS, the meaning of the first four numbers 1, 1, 3 and 1 is as follows: the root node has zero attribute, the first, second and third children of the root node have 0, 2 and 0 attributes, respectively. Zero stands for no children and the number 4 means that the number of attributes is 3 or more.

The other simplifying procedure, the *left-ordered* representation, was developed to avoid combinatorial explosion. In the case of data-driven XML documents, the order of the child nodes under the parent is arbitrary. This freedom in the structure means that the very same document can appear in different forms. Left-ordering makes the order of the child nodes well determined. We call a representation 'left-ordered' if it has the following properties: a node is on the left of its own sibling if this node has more children than the sibling or, in the case of equality, it has more attributes than its sibling. This representation sacrifices the original ordering of XML elements to make the structure unambiguous. This simplification may have to be avoided for documents representing text data (i.e. for a document-driven XML approach), provided that rule-based pre-filtering cannot take care of this problem.

With these definitions, we have structure-dependent probabilistic variables. Every document is mapped to a configuration of $(X_1,..., X_n)$. With this construct, or one similar to it, distributions of document structures can be created, provided that sets of examples are available. In our work, different sets of 'training' documents were converted into this form and learning methods were applied to estimate the probability distributions sampled by the training set.

Probabilistic tree model

The task of representing the discrete $P(X^n) = P(X_1,..., X_n)$ probability distribution has its own problems. The main challenge is avoiding the curse of the dimensionality problem. The number of probabilities of all possible

configurations grows exponentially with the number of variables and, therefore, the number of training samples required for building an accurate table grows exponentially too. Obviously, we need a probabilistic model that represents distributions subject to the constraints of limited resources. It is also desired that the loss of accuracy be minimised by the choice of our model.

Factorised representation with optimal structure

Chow and Liu [15] suggested a method that learns and represents $P(X_1,...,X_n)$ in a factorised form as a product of pairwise joint probability distribution functions. This approach is effective for distributions that have strong pairwise correlations between variables. Their algorithm can be summarised as follows (for details see [15, 16, 17]):

- Compute the marginal distributions, $P_i(X_i)$, for every variable X_i ($i=1,..., n$)
- Compute the pairwise marginal distributions, $P_{ij}(X_i, X_j)$, for every pair of variables, X_i and X_j ($i, j=1,..., n$)
- Compute the mutual information I_{ij} for every variable pair. I_{ij} depends only on the marginal distributions:

$$I_{ij} = \sum_{x_i,x_j} P_{ij}(x_i,x_j) \log\left(\frac{P_{ij}(x_i,x_j)}{P_i(x_i)P_j(x_j)}\right) \quad (1)$$

- Build a complete graph whose vertices are the variables and the edges are weighted by I_{ij}
- Find a maximum weight spanning tree (MWST) on this graph (e.g. [18])
- The distribution function, $P(X)$, can be approximated by the following factorised form:

$$T(X) \doteq \frac{\prod_{(uv)\in E} P_{uv}(X_u, X_v)}{\prod_{X_u\in V} P_u(X_u)} \quad (2)$$

where V denotes the set of probabilistic variables and E denotes the set of edges of the MWST.

An equivalent model using conditional probabilities can be written in the following form:

$$T(X) = \prod_{X_u\in V} P_{u|pa(u)}(X_u|X_{pa(u)}) \quad (3)$$

where $pa(u)$ represents the parent of the vertex u in a directed tree. In this case, we must have a root vertex and directions assigned to the edges, starting from the root. With these definitions, $T(X)$ is an approximation of the true $P(X)$ distribution.

The memory needed to store a distribution in this form depends on the number of joint probability tables. In turn, the required space depends linearly on the number of variables (assuming equal or similar cardinalities for the variables). Computation depends quadratically on both the number of variables and the possible values of each variable.

Simplified training of tree models on XML documents

The Chow-Liu algorithm computes a factorised distribution function that is the 'closest' to the experienced distribution in the Kullback-Leibler (KL) sense.

Computing mutual information, I_{ij}, for all pairs of variables is rather time consuming. It requires four nested loops; two loops are needed to sweep over all pairs of variables and another two loops are necessary for the marginalisation of these variables using the two-dimensional probability tables. In the case of XML documents, we can simplify this procedure considerably by exploiting the natural tree structure of such documents. The price to pay is that this tree structure may not correspond to the optimal factorisation of $P(X)$. However, it is rational to expect strong correlation between a tag and its parent. The tree structure can be sub-optimal, for example, when correlations between remote tags is stronger than the correlation between one of these tags and its parent. We have run many tests on XML sets and found that the $T(X)$ distribution, when factorised according to the natural structure, is a good approximation of $P(X)$.

Probabilistic XML classifier

Consider n different XML data sets, $D_1,..., D_n$. We assume that all known strict rules about data in the data sets have been applied and that the data set has been reduced accordingly. In turn, the data within each data set, to our best knowledge, are unbiased samples of the distribution belonging to that data set. We assign a probabilistic tree model to each data set ($T_1,..., T_n$). These probabilistic tree models will be used for classification. When new data (a new XML document) are received from an unknown source, it is measured by each of the $T_1,..., T_n$ models. The source of the data can be identified as the class of the model, which returns the greatest probability. Alternatively, the membership of the document can be established by normalising the obtained probability values to one. Membership could be used for further analysis, e.g. for novelty or fault detection when all probabilities are low. A trained model produces probability estimations of different possible memberships.

Novelty detection

Novelty detection is a special form of classification. Here, one has to make a decision, based on a probability value estimated by a model, whether the given sample belongs to any of the sets or not. For decision-making, we need a suitable acceptance threshold on probability values. Determining any threshold is a difficult task, for the following reasons:

1. The model may provide probabilities for 'familiar' data spanning several orders of magnitude, even for

documents which are similar to those in the training set (this will be our case)
2. The number of correctly identified samples needs to be maximised, whereas
3. The number of misclassified samples should be minimised

The optimal threshold that gives a good balance between the second and third conditions is usually strongly task-dependent.

Let $E_r(\theta)$ denote the incorrectly rejected samples and $E_a(\theta)$ denote the incorrectly accepted samples, where θ denotes the value of the threshold. The cumulative error can be defined by:

$$E_c = \alpha_r E_r + \alpha_a E_a \quad (4)$$

where α_r and α_a are constants. In practice, the importance of incorrectly accepted or rejected samples can be balanced by choosing appropriate coefficients. We shall examine this cumulative error on XML documents for the case where $\alpha_r = \alpha_a = 1$.

Data sets

In this section, we describe the data sets used to study the accuracy of the probabilistic tree method in classification and novelty detection experiments. Artificial data sets and data sets from the Internet were both used in these tests. Although the proposed method is intended to be used in 'real-world' applications, artificial data sets still have greater importance when robustness or accuracy are investigated. In the case of generated data sets, we have direct influence on the variance and noise of every relevant property such as depth, branching ratio or number of attributes. When using downloaded 'real-world' data, these variances are 'wired' into the data sets. Based on these considerations, our investigations were weighted towards artificial data sets.

Internet data sets

With the increasing popularity of XML structure, more and more sites offering information in this form can be found. These documents can be accessed via two basic ways:

- The site stores the data in a hidden database, e.g. in binary form. One cannot access the files of the database directly but only via an interface (e.g. by using a query form on a web page) and the requested information is returned in XML form. These data sets are often generated by templates and have identical structure. In this case, the distribution coalesces to the trivial distribution. Although an exact match of these structures can be recognised by learned distribution functions, this is a typical case when syntax-based validation is much faster and should be utilised. We have found that template-based data sets are dominant on the Web today, especially on commercial sites.
- The other site type offers native XML databases on the Internet. Normally, universities and non-profit organisations follow this approach. In this case, one can have direct access to the data files via ftp or http protocols and the documents can be collected automatically.

Two data sets were collected for our experiments, one from the Astronomy Data Center [19] and one on chemical test data [20]. These differed in their varieties considerably. XML files in [20] all have an identical structure. This collection is an example of template-based data sets. The files in [19] each have a different structure, but they all follow a recognisable distribution.

Artificial data sets

To test our method, artificial XML-like data sets were used. A data set was generated in two steps. Firstly, a prototype document is created and then the other members of the data set were created by sweeping over its nodes with perturbation operators simulating real variations.

A parameterised set of 'prototype documents' was created (Fig. 3). Individual 'documents' were distorted or perturbed versions of this prototype. In fact, the parameters of the individual documents were taken from a distribution around those of the prototype document. The parameters subject to randomisation in the class generation procedure were as follows:

- Depth of the tree
- Lower and upper bounds of branching ratio at a node (i.e. the minimum and maximum number of children of a node)
- Probability distribution of the number of children between the bounds
- Probability distribution on the number of (XML) attributes at each node

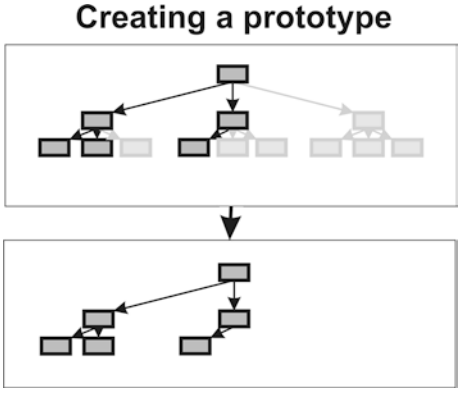

Fig. 3 'Prototype' generated randomly. The *upper* part depicts the generation procedure. The result, i.e. the 'prototype', is shown in the *lower* part

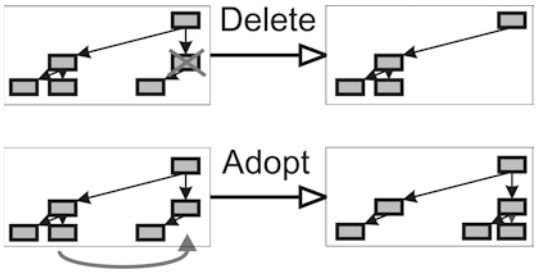

Fig. 4 The effects of the 'delete' and 'adopt' operators. Delete erases tags and all of its descendants randomly. Adopt replaces tags and their descendants randomly among siblings

Table 1 Generative parameters of artificial data sets. The first section shows the parameters used for generating a base document. The second section contains the parameters used to perturb the base document (see text for details). The last section shows the parameters of the distribution of the attribute numbers

	Dat1	Dat2	Dat3	Dat4	Dat5
Size of the data set	1001	1001	2001	257	431
Sizes of XML tree structure					
Upper bound of branching number	4	4	5	5	5
Upper bound of depth	4	4	5	5	5
Generated depth of 'prototype'	4	4	4	4	5
Operator probabilities					
Uniform distribution[a]					
Delete probability P_D	0.1	0.1	0.1	0.2	0.1
Adopt probability P_A	0.2	0.2	0.1	0.2	0.1
Increase of $P_D$by depth	0.05	0.05	0.05	0	0.03
Increase of $P_A$by depth	0.05	0.05	0.05	0	0.03
Attribute probability data					
Sampled truncated Gaussian[b]					
Mean of the Gaussian	4	2	2	3	2
Standard deviation	1	1	1	1.5	0.5
Probability of updating attributes	0.3	0.3	0.3	0.3	0.3
Upper bound for attribute number	6	6	3	5	6

[a]These rules were not applied for the root
[b]Gaussian distribution is sampled on discrete values restricted to positive integers from below and by the upper bound from above

To model the most common variations in structuring a specific piece of information, and also to model errors that may occur in XML documents, two operators were applied (Fig. 4):

1. The *deletion operator* deletes tags (and their descendants) randomly, simulating missing information
2. The *adoption operator* reorders tags (and its descendants) amongst the siblings (other children of the parent), simulating mistypings or partial reorganisations in the data structure. For example, 'maiden name' could be placed under the 'personal data' tag or under the 'name' tag

The probability of applying these operators is a function of depth. Deletion of the root node, or many nodes at higher levels, needs to be avoided to make a higher variety of databases. We usually apply larger probabilities for deeper nodes of the tree. Generating many files with these operations from the base document by following a particular probability distribution was considered as a class of documents. Details of the parameters of the generation procedure and the data sets are given in Table 1.

Experiments and results

Computing probability distributions of generated data sets

Here, we will demonstrate that structure-based probability distributions learned by the tree model can be characteristic to the respective data, and that a few sample XML files are sometimes sufficient for reasonably good estimations. We applied the following method to identify data sets. A data set from a specified probability distribution was separated into two parts; a training set (for data set, Dat*i*, this set is denoted as T_{Dati}) and a control set (sometimes called a test set and denoted by C_{Dati}). The probabilistic tree model was trained according to the training set. The result, for example, for data set i is the ith tree model. Any tree model could be used to compute probabilities for samples from any of the control sets. The results for two data sets, Dat1 and Dat2, are shown in Fig. 5. The corresponding training and control sets are T_{Dat1}, C_{Dat1}, T_{Dat2} and C_{Dat2}.

The histogram of probability distributions determined by the jth tree model on the ith training and test data set are denoted by $H_j(T_{Dati})$ and by $H_j(C_{Dati})$, respectively.

According to our experiments, a small amount of training data is sufficient for the model to give similar distributions for the training and control sets belonging to the same class. Probability values covered several orders of magnitude. Histograms on data sets belonging to different prototypes can look very different. The histogram of one model on the other document class has much lower probabilities, as shown in Fig. 5.

Results of classification experiments

Different data sets with different characteristic structures can be identified by using pre-trained tree models. The classification experiments were performed on data sets from three distinct sources. Computational experiments differed in the number of training samples chosen from the three data sets. Three probabilistic tree models were trained on the three data sets respectively. The three models assigned probability values to the remaining elements of the three data sets and classification followed the probability estimation; the class of the data was determined as the class of the model that assigned the greatest probability estimate to that data. Several

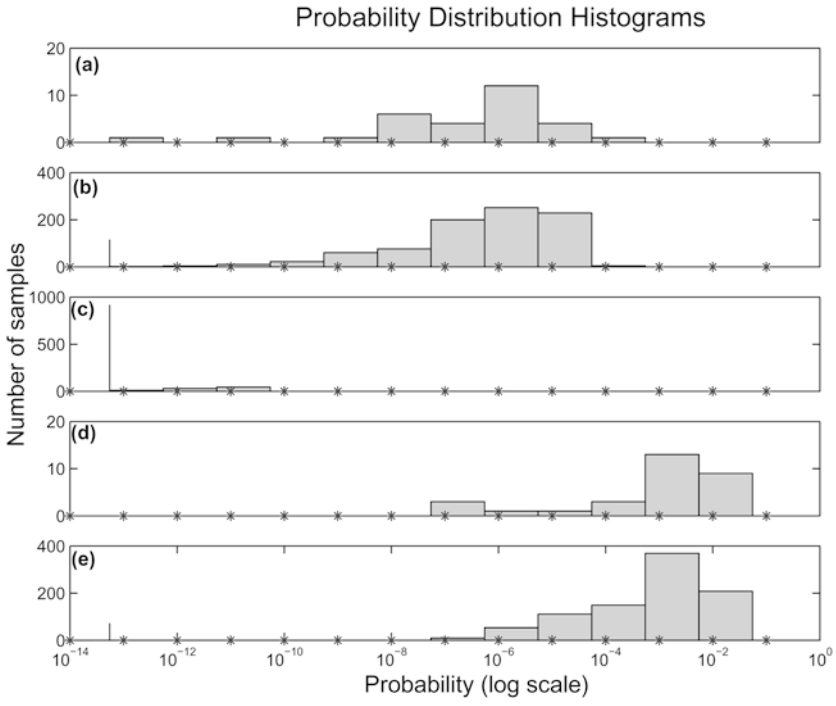

Fig. 5a–e Results of the probabilities approximated by the probabilistic tree models. **a** Histogram of model 1 computed on its own training samples from the first data set, $H_1(T_{Dat1})$. **b** Histogram of model 1 on unseen samples from the first data set, $H_1(C_{Dat1})$. **c** Histogram of model 1 on samples from the second data set, $H_1(C_{Dat2})$. **d** Histogram of model 2 computed on its own training samples from the second data set, $H_2(T_{Dat2})$. **e** Histogram of model 2 on unseen samples from the second data set, $H_2(C_{Dat2})$. Model parameters: training set size = 30 samples, branching ratio = 4, depth of tree = 4, first data set = Dat1, second data set = Dat2 (for other parameters of these artificial data sets, see Table 1)

experiments were conducted covering different data sets, the number of elements in the training sets and the parameters of the basic constrained structures. Representative results are tabulated in Table 2.

Results of novelty detection experiments

Novelty detection experiments were performed on two data sets. One of the data sets was used to train a tree model. Another data set was chosen as a set of novel samples. In some of the experiments, an artificial or natural data set was used as the novelty set, whereas in some other experiments, we used a special data set named 'garbage'. This 'garbage' data set contained tree structures generated independently with random parameters instead of using the deletion and adoption operators. 'Garbage' simulates a larger variance of 'novel' documents, which may come from arbitrary independent sources of the Web.

Figure 6 shows the number of unrecognised (non-accepted) documents, the number of misclassified (accepted) documents and the cumulated error (as defined in Eq. 4) as a function of the threshold value. The cumulated error shows a sharp minimum when the probabilities are depicted on a logarithmic scale. This sharp threshold was typical for the cases we studied. The optimal threshold value is, however, database-dependent. Table 3 summarises the results of several experiments run on different data sets with various 'novelty sets'. Optimal threshold values for the different cases are also tabulated in Table 3.

Table 2 Summary of classification experiments, showing the different data sets used as different classes and model parameters in the experiments. Set *i* is the *i*th data set used to train the tree model. ADC = Astronomy Data Center. ECML = Examples on Chemical Markup Language

Set 1	Set 2	Set 3	Branching ratio[a]	Depth of BCS[b]	No. of training samples	No. of samples	Misclass'd samples[c]	Correct classif.[d]
Dat1	Dat2	Dat3	3	4	10	3973	775	0.804
Dat1	Dat2	Dat3	3	4	30	3913	225	0.942
Dat1	Dat2	Dat3	3	4	100	3703	65	0.982
Dat1	Dat2	Dat3	2	5	30	3913	161	0.959
Dat1	Dat4	Dat5	3	4	30	1599	250	0.843
Dat2	Dat3	Dat5	3	4	30	3343	90	0.973
Dat4	Dat3	Dat2	4	2	30	3169	64	0.980
Dat4	Dat3	Dat2	4	3	30	3169	85	0.973
ADC	ECML	Dat1	3	4	30	3341	162	0.952
ADC	ECML	Dat1	4	3	30	3341	25	0.993

[a]Branching ratio of the basic constrained structure
[b]Depth of the basic constrained structure
[c]Number of misclassified samples
[d]Ratio of correctly identified samples

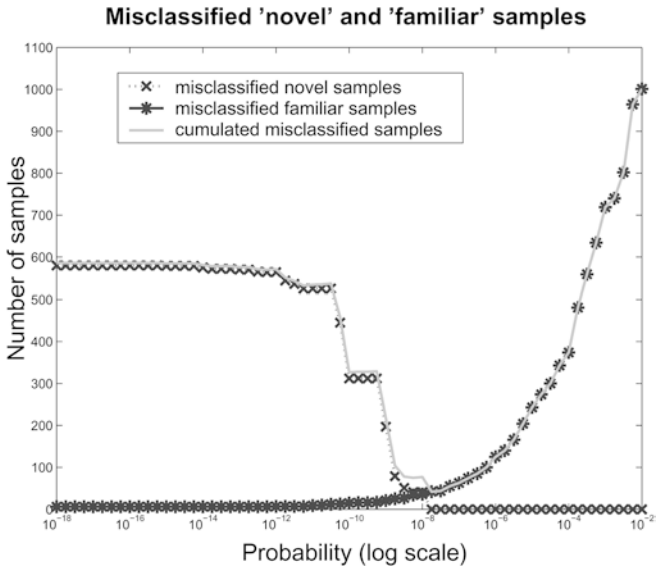

Fig. 6 Novelty detection vs threshold value. The number of errors ('misclassified familiar samples' = misclassified samples labelled as 'novel', 'misclassified novel samples' = samples belonging to the 'novel' set classified as within class) as a function of threshold. Model parameters: size of training set = 150 samples, branching ratio = 3, depth of tree = 4, 'familiar' data set = Dat2, 'novelty' data set = Dat3. (For parameters of Dat2 and Dat3, see Table 1)

Discussion

Selecting the learning method for XML documents

When working with text or XML documents, the data can appear in various forms. A two-stage process is necessary here. Firstly, the data is transformed into a mathematically treatable form. In most cases, this means that a document is assigned a point in a suitably defined vector space. At the second stage, general machine learning methods could be used to perform validation, classification, novelty detection and other tasks. The first stage determines the type of information the method will work on. The second stage is responsible for performing the particular task. Obviously, if the machine learning method is carefully chosen so as to be consistent with the task, the quality of the method will depend mainly on the first step.

In the first step, we create a vector representation of the XML document tree. After this, a probabilistic tree method [15] is applied to extract the probabilistic information from the representation. The chosen method has good generalisation properties, requires a small amount of training data and can adapt to changes made in XML documents as information systems evolve. Applying other methods, such as SVM or other known methods, on the vector representation would also have been possible. As an early design decision, however, we opted for the tree learning system because (1) it is straightforward to use with XML documents due to their tree structure and (2) it is advantageous in multi-class decision problems as well.

This architecture has an important feature; it has modules capable of providing more refined outputs than a simple binary 'accept' or 'reject'. This is an attractive property because a good classification scheme should be able to work with 'symptoms' and 'sicknesses', where each 'sickness' has a probability distribution of 'symptoms', but we are given the symptoms and have to deduce the 'sickness' (this is the diagnostic process) to choose the 'remedial tools' (the means of error correction). Such methods can give probability distributions on document classes, i.e. a particular document can be a member of several document classes with different probabilities. In addition, these modules can work in a parallel manner, which, in multi-processor systems, is a very fruitful approach. Every module acts as an expert, processes the document independently and provides an output. The outputs are then combined into a final distribution over document classes. Each and every module can reduce the uncertainty in the final distribution.

Table 3 Summary of the data sets and model parameters used in the novelty detection experiments

Training set[a]	Novelty set[b]	Branching ratio[c]	Depth of BCS[d]	Sample number[e]	Threshold[f]	Recn'd. TrainSmpls[g]	MisClass'd TrainSmpls[h]	Recn'd. NovSmpls[i]	MisClass'd NovSmpls[j]	Min. Error[k]	Opt. Perform[l]
Dat1	Dat2	3	4	2002	1.78×10^{-8}	959	42	1001	0	42	0.979
Dat2	Dat3	3	4	3002	3.16×10^{-6}	957	44	1001	0	44	0.985
Dat4	Dat2	3	4	1258	$2{,}78\times10^{-6}$	212	45	1001	0	45	0.964
Dat4	Dat5	3	4	688	$1{,}78\times10^{-5}$	185	72	423	8	80	0.883
Dat4	Dat5	3	5	688	5.62×10^{-18}	230	27	367	64	91	0.867
Dat2	Garbage	3	4	1801	5.62×10^{-6}	954	47	678	122	169	0.906
ECML	Garbage	3	5	1074	5.62×10^{-18}	246	1	798	2	3	0.997
ADC	Garbage	3	5	2983	0	2183	0	800	0	0	1

[a]The data set used to train a tree model
[b]The data set used as novelty samples
[c]Branching ratio of the basic constrained structure
[d]Depth of the basic constrained structure
[e]Overall number of samples in the training and novelty sets
[f]Optimal threshold of probabilities for accepting/rejecting samples
[g]Recognised training samples
[h]Unrecognised training samples
[i]Recognised novel samples
[j]Novel samples classified erroneously
[k]Minimised misclassification. The minima of the sum of unrecognized training samples and misclassified novel samples as a function of threshold
[l]Optimal performance at the best threshold

We have mentioned two examples for complex XML processing systems. In practical applications, any combination of a sequential 'validator' and a parallel probability processor can be used to form a hybrid system. These complex systems can be designed or optimised to be fast, accurate and adaptive. We think that our structure-based method is an important contribution to such systems.

The Internet and the methods of information retrieval from the data available on the Internet develop quickly and undergo fast evolution. Classification and clustering according to structural similarities of documents, especially XML documents, are clearly in need. Research on this field has just started, and much of the results are available only on the Internet. One approach defines a metric to quantify structural similarity between an XML document and a DTD [7]. This method, for example, can be used as a pre-filtering tool; if the document passes this pre-filtering then it can be converted into a left-ordered representation. Another method emphasises the importance and the potential of similarity queries [21]. It is expected that combinations of such methods, and similar, will be used in the near future.

We have tried to consider a larger set of validation tools, which may include probabilistic inference components about the structure of the XML document (see Fig. 1). The potential of the probabilistic tree model was studied in this paper. This model has the following advantages: (1) it can be fast, (2) it can work on a small number of training samples and (3) training can be fast. Therefore, we think that the probabilistic tree algorithm can be seen as a fast filtering algorithm for any database made of directed acyclic graphs or Markov random fields. In other words, one may need to select candidates of high probabilities and filter out candidates of low probabilities in many different problems, which can be formulated as directed acyclic graphs. Candidates with non-vanishing probabilities may undergo further analysis. Note, that there is no unique answer for such systems and errors may (and will) occur. The first point is that filtering may speed up classification with minor compromises on the error rate. The second point is that tree-based representation can be extended to a mixture of trees (MoT), which is a more general probabilistic model on tree structures. If extended to MoT, present limitations of our algorithm will disappear; MoT is not restricted in its representational power [16]. However, learning time and probability estimation of MoT may depend on several parameters of the database, and careful choice of the classification scheme might be necessary. Other possibilities could be of use here. For example, the SVM classifiers, which are fast but may require extensive training.

Size of the training set and novelty detection

According to Table 2, a small number of training samples may provide good results. Training sample sizes of 10, 30 and 100 gave rise to correct classification percentages of 80.4%, 94.2% and 98.2%, respectively. That is, the effort on collecting training samples can be stepwise, and it turns out that reasonable precision can be achieved by a small number of training samples.

XML documents collected from Astronomy XML Resources (ADC) [19] and from Examples on Chemical Markup Language (ECML) [20] have different stochastic properties. This is shown clearly in the novelty recognition task. Novelty detection is one of the most important problems of classification and recognition for systems under continuous development. Careful investigation is needed for at least three cases:

1. The XML document has a reasonable similarity to the documents we are collecting and should undergo further analysis to improve rejection and misclassification rates. It is also possible that further analysis is needed to fix the document under investigation
2. The XML document is novel, it is a not-yet-seen structure which, by accident, may look similar to the examples we are collecting and we may conclude that we do not need it
3. The XML document is novel, it should be collected, but it has a structure not experienced before

The last two cases need further analysis. In the case of the second point, we need to improve our *decision surface* for a better rejection rate. In the case of the third point, we need to analyse, collect and learn this novel type of document. In fact, learning means the recognition, analysis and classification of the novel information.

We have found that the threshold for optimal novelty detection is not too sharp (note the logarithmic scale of Fig. 6). Nevertheless, there is strong dependence on this threshold, which means to us that training and the adaptation of this threshold should be maintained for databases subject to quick changes.

Summary

We can summarise our results as follows: (1) the classification of graphs can be efficient for trees, (2) the probabilistic tree description fits XML documents in a natural fashion; our studies show that the probabilistic tree method can be an optimal compromise for graph classification. This last point can be justified as follows; trees with increasing complexities can be tested to optimise resource and time requirements on learning and, for a given performance criterion, trees can be a flexible means for (hierarchical) pre-filtering, and the probabilistic tree method can be extended to mixture of trees to give a general form of representation of probability distributions [16]. We note that structural information represented by a tree can be efficient in novelty detection, a problem of growing importance in the fast changing world of distributed databases.

Originality and contributions

Owing to the rapid evolution of network technologies, a huge amount of electronic data is available on the Internet in both HTML and XML formats. These data have no rigid structure and are often referred to as semi-structured data [20]. There is an increasing need for automated methods to extract the useful information, and particularly to discover rules or patterns in a large collection of semi-structured data. Recent research on information extraction from structured documents has focused primarily on grammatical inference methods [19].

In this paper, we take a structure-based probabilistic approach. The probabilistic tree method (Chow and Liu) is transcribed and shaped to fit the characteristic needs of XML document classification on the Internet. This method is tested for classification and also for novelty detection on XML documents, both those generated from several probability distributions as well as sample documents downloaded from the Internet. The efficiency of the method is demonstrated for the recognition (or rather, for the probability estimation) of XML patterns. The algorithm can serve as an add-on technology. It is anticipated that a combination of this and similar methods will be included into future wrappers (e.g. HTML Tidy http://www.w3.org/People/Raggett/tidy/) or classification systems.

1. Abiteboul S, Buneman P, Suciu D (2000) Data on the web: From relations to semistructured data and XML. Morgan Kaufmann, San Francisco, CA
2. Kosala R, Van den Bussche J, Bruynooghe M, Blockeel H (2002) Information extraction in structured documents using tree automata induction. In: Elomaa T, Mannila H, Toivonen H (eds) Principles of data mining and knowledge discovery: Lecture notes in computer science, vol 2431, 6th European conference, PKDD 2002, Helsinki, Finland, August 2002, pp 299–310
3. Cohen WW (1999) Recognizing structure in web pages using similarity queries. AAAI/IAAI, pp 59–66

About the authors

György Hévízi received his MSc degree in Physics in 2000. Since then, he has been continuing two graduate studies, one in Physics at the University of Szeged and one in Information Processing at the Eötvös Loránd University.

Tamás Marcinkovics received his MSc degree as a programming mathematician at the Eötvös Loránd University in 2001. Since then, he has been affiliated with IQSoft, Hungary.

András Lőrincz has been doing research on pattern recognition, brain modelling and adaptive information systems. He has been leading the Neural Information Processing Group of ten PhD students at the Eötvös Loránd University. He received his PhD in 1978 from ELU and became a professor at the University of Szeged in 1998. His background is in theoretical and experimental physics.

Acknowledgements This work was supported by the Hungarian National Science Foundation (Grant OTKA 32487) and by EOARD (Grant F61775–00-WE065). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the European Office of Aerospace Research and Development, Air Force Office of Scientific Research or the Air Force Research Laboratory.

References

1. Berners-Lee T, Hendler J, Lassila O (2001) The semantic web. Sci Am 284(5):34–43
2. Abiteboul S, Buneman P, Suciu D (2000) Data on the web: From relations to semistructured data and XML. Morgan Kaufmann, San Francisco, CA
3. Kosala R, Van den Bussche J, Bruynooghe M, Blockeel H (2002) Information extraction in structured documents using tree automata induction. In: Elomaa T, Mannila H, Toivonen H (eds) Principles of data mining and knowledge discovery: Lecture notes in computer science, vol 2431, 6th European conference, PKDD 2002, Helsinki, Finland, August 2002, pp 299–310
4. Muslea I, Minton S, Knoblock CA (2001) Hierarchical wrapper induction for semistructured information sources. Auton Agent Multi-Ag 4(1–2):93–114
5. Sahuguet A, Azavant F (2001) Building intelligent web applications using lightweight wrappers. Data Knowl Eng 36(3):283–316
6. Lee D, Chu WW (2000) Comparative analysis of six XML schema languages, SIGMOD Record 29(3):76–87
7. Bertino E, Guerrini G, Mesiti M, Rivara I, Tavella C (2001) Measuring the structural similarity among XML documents and DTDs
8. Miller GA (1995) WordNet: a lexical database for English. Commun ACM 38(11):39–41
9. Jacquin DC (2001) Indexing a web site with a terminology oriented ontology
10. Jutten C, Herault J (1991) Blind separation of sources. Part I: An adaptive algorithm based on neuromimetic architecture. Signal Process 24:1–10
11. Comon P (1994) Independent component analysis – A new concept? Signal Process 36:287–314
12. Cardoso JF, Laheld B (1996) Equivalent adaptive source separation. IEEE T Signal Proces 44:3017–3030
13. Bell AJ, Sejnowski TJ (1995) An information-maximization approach to blind separation and blind deconvolution. Neural Comput 7:1129–1159
14. Amari SL, Cichocki A, Yang HH (1996) A new learning algorithm for blind signal separation. Advances in neural information processing systems. Morgan Kaufmann, San Mateo, CA, pp 757–763
15. Chow CK, Liu CN (1968) Approximating discrete probability distributions with dependence trees. IEEE T Inform Theory 14:462–467
16. Meila M, Jordan MI (2000) Learning with mixtures of trees. J Mach Learn Res 1:1–48
17. Meila-Predoviciu M (1999) Learning with mixtures of trees. PhD thesis, Massachusetts Institute of Technology, January 1999
18. Cormen TH, Leiserson CE (1990) Introduction to algorithms. MIT Press, Cambridge, MA
19. ADC XML resourceshttp://xml.gsfc.nasa.gov
20. Examples of CMLhttp://www.xml-cml.org/examples
21. Cohen WW (1999) Recognizing structure in web pages using similarity queries. AAAI/IAAI, pp 59–66