A TRANSACTIONAL APPROACH TO ASSOCIATIVE XML CLASSIFICATION BY CONTENT AND STRUCTURE

Gianni Costa, Riccardo Ortale and Ettore Ritacco
ICAR-CNR, Via P. Bucci 41C, 87036 Rende (CS), Italy

Keywords: XML mining, XML transactional modeling, Associative classification.

Abstract: We propose XCCS, which is short for XML Classification by Content and Structure, a new approach for the induction of intelligible classification models for XML data, that are a valuable support for more effective and efficient XML search, retrieval and filtering. The idea behind XCCS is to represent each XML document as a transaction in a space of boolean features, that are informative of its content and structure. Suitable algorithms are developed to learn associative classifiers from the transactional representation of the XML data. XCCS induces very compact classifiers with outperforming effectiveness compared to several established competitors.

1 INTRODUCTION

XML is a popular model for data representation, that allows to organize textual content into (possibly irregular) logical structures.

The supervised classification of XML data into predefined classes consists in learning a model of the structural and content regularities (observed across a set of pre-classified XML documents), that discriminate each individual class. The resulting classifier can, hence, predict the class of a previously unseen XML document from the same applicative domain, by looking at its structure and content.

A wide variety of approaches to XML classification have been proposed in the literature, including (Theobald et al., 2003; Yi and Sundaresan, 2000; Garboni et al., 2006; Knijf, 2007; Zaki and Aggarwal, 2006). These efforts can be divided into two major categories. One family of approaches uses only the structural information of XML data in classifier induction and class prediction, such as in (Garboni et al., 2006; Knijf, 2007; Zaki and Aggarwal, 2006). An inherent limitation of such approaches is the inability at discriminating the classes, when all of the available XML documents share an undifferentiated structure. Another family of approaches, such as the ones in (Theobald et al., 2003; Yi and Sundaresan, 2000), performs a more sophisticated separation of the classes, by considering both the content and structural information of XML data. Unfortunately, despite their effectiveness, these approaches do not provide explicative classification models, i.e., concise and human-intelligible summarizations of the content and structural regularities that discriminate the individual classes. Such classification models have the potential to offer an in-depth and actionable understanding of the relevant properties of very large corpora of XML data and, hence, are of great practical interest in all those settings in which XML classification is preliminarily performed to enable more effective and efficient XML search, retrieval and filtering.

In this paper, we propose XCCS, a new approach to XML Classification by Content and Structure, that relies on solid and well-understood foundations. XCCS performs associative classification on the available XML data to induce an easily interpretable and highly expressive predictive model. The latter is a compact set of rules, which discriminate the generic class from the others by means of content and structural regularities, that frequently occur in the class and are positively correlated with the class itself.

We identify suitable features of the XML documents, that are informative of their content and structure, and represent each XML document as a transaction in the resulting feature space. Additionally, we design algorithms to perform associative classification on the transactional representation of the XML data. The devised algorithms handle skewed class distributions, that are often encountered in the XML domain. To the best of our knowledge, XCCS is the first approach that borrows the advantages of associative classification, i.e., a high degree of both interpretabil-

Costa G., Ortale R. and Ritacco E..
A TRANSACTIONAL APPROACH TO ASSOCIATIVE XML CLASSIFICATION BY CONTENT AND STRUCTURE.
DOI: 10.5220/0003662401040113
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval* (KDIR-2011), pages 104-113
ISBN: 978-989-8425-79-9

ity and expressiveness (that are well known and studied in the literature) coupled with a robust effectiveness. As a further contribution, the latter is investigated by comparatively evaluating XCCS over several XML corpora. Empirical evidence shows that XCCS scales to induce very compact classifiers with outperforming effectiveness from very large XML corpora.

The paper proceeds as follows. Section 2 introduces notation and preliminaries. Section 3 discusses the XCCS framework. Section 4 presents the results of a comparative evaluation of several classifiers induced within XCCS. Finally, section 5 concludes and highlights future research.

2 PRELIMINARIES

We introduce the notation used throughout the paper as well as some basic concepts. The structure of XML documents without references can be modeled in terms of *rooted, labeled trees*, that represent the hierarchical relationships among the document elements (i.e., nodes).

Definition 2.1. ***XML Tree.*** *An XML tree is a rooted, labeled tree, represented as a tuple* $\mathbf{t} = (r_\mathbf{t}, \mathbf{V_t}, \mathbf{E_t}, \lambda_\mathbf{t})$, *whose individual components have the following meaning.* $\mathbf{V_t} \subseteq \mathbb{N}$ *is a set of nodes and* $r_\mathbf{t} \in \mathbf{V_t}$ *is the root node of* $\mathbf{t}$, *i.e. the only node with no entering edges.* $\mathbf{E_t} \subseteq \mathbf{V_t} \times \mathbf{V_t}$ *is a set of edges, catching the parent-child relationships between nodes of* $\mathbf{t}$. *Finally,* $\lambda_\mathbf{t} : \mathbf{V_t} \mapsto \Sigma$ *is a node labeling function and* Σ *is an alphabet of node tags (i.e., labels).* □

In the above definition, the elements of XML documents and their attributes are not distinguished: both are mapped to nodes in the XML tree representation. Hereafter, the notions of XML document and XML tree are used interchangeably.

Let $\mathbf{t}$ be a generic XML tree. Nodes in $\mathbf{V_t}$ divide into two disjoint subsets: the set $\mathbf{L_t}$ of *leaves* and the set $\mathbf{V_t} - \mathbf{L_t}$ of *inner nodes*. An inner node has at least one child and contains no textual information. A leaf is instead a node with no children, that can contain only textual information.

A root-to-leaf path $p_l^{r_\mathbf{t}}$ in $\mathbf{t}$ is a sequence of nodes encountered in $\mathbf{t}$ along the path from the root $r_\mathbf{t}$ to a leaf node l in $\mathbf{L_t}$, i.e., $p_l^{r_\mathbf{t}} =< r_\mathbf{t}, \ldots, l >$. Notation $\lambda_\mathbf{t}(p_l^{r_\mathbf{t}})$ denotes the sequence of labels that are associated in the XML tree $\mathbf{t}$ to the nodes of path $p_l^{r_\mathbf{t}}$, i.e., $\lambda_\mathbf{t}(p_l^{r_\mathbf{t}}) =< \lambda_\mathbf{t}(r_\mathbf{t}), \ldots, \lambda_\mathbf{t}(l) >$. The set $paths(\mathbf{t}) = \{p_l^{r_\mathbf{t}} | l \in \mathbf{L_t}\}$ groups all root-to-leaf paths in $\mathbf{t}$.

Let l be a leaf in $\mathbf{L_t}$. The set $terms(l) = \{\lambda_\mathbf{t}(p_l^{r_\mathbf{t}}).w_1, \ldots, \lambda_\mathbf{t}(p_l^{r_\mathbf{t}}).w_h, \lambda_\mathbf{t}(p_l^{r_\mathbf{t}}).\varepsilon\}$ is a model of the information provided by l. Elements $\lambda_\mathbf{t}(p_l^{r_\mathbf{t}}).w_i$ (with $i = 1 \ldots h$) are as many as the distinct term stems in the context of l and seamlessly couple content information with its structural context. Therein, w_i is some term stem (obtained in the first step of the preprocessing in subsection 4.2) and $\lambda_\mathbf{t}(p_l^{r_\mathbf{t}})$ acts as a prefix specifying the location of l within the XML tree $\mathbf{t}$, that allows to distinguish the occurrences of w_i in the context of distinct leaves. The unique element of the type $\lambda_\mathbf{t}(p_l^{r_\mathbf{t}}).\varepsilon$ is instead informative only of the location of l within $\mathbf{t}$: ε indicates the null string. $\lambda_\mathbf{t}(p_l^{r_\mathbf{t}}).\varepsilon$ still provides some (purely-structural) information on l when the leaf does not contain textual information.

Leaf terms and their prefixes are chosen as informative features of the XML data, with which to separate the classes of XML trees. Henceforth, for readability sake, we will write p instead of $\lambda(p)$ to mean the prefix of a term stem w.

Definition 2.2. ***XML Feature.*** *Let* $\mathbf{t}$ *be an XML tree. A prefixed term (stem)* $p.w$ *is said to be a* feature *of* $\mathbf{t}$ *(or, equivalently,* $p.w$ *occurs in* $\mathbf{t}$*), denoted as* $p.w \preceq \mathbf{t}$, *if the following two conditions hold. First, there exists a leaf* $l \in \mathbf{L_t}$ *and, hence, a path* $p_l^{r_\mathbf{t}} \in paths(\mathbf{t})$ *such that* $\lambda_\mathbf{t}(p_l^{r_\mathbf{t}}) = p$. *Second,* $p.w \in terms(l)$. □

Assume that $\mathcal{S} = \{p.w | \exists \mathbf{t} \in \mathcal{D}$ such that $p.w \preceq \mathbf{t}\}$ is a suitable selection of features from the XML trees in $\mathcal{D}$. $\mathcal{S}$ identifies an expressive feature space, in which to perform the induction of models for effective XML classification.

Let $\mathcal{D} = \{\mathbf{t}_1, \ldots, \mathbf{t}_N\}$ be a training database (or, equivalently, a forest) of N XML trees, each of which is associated with one label from the set $\mathcal{L} = \{c_1, \ldots, c_k\}$. Our approach to XML classification can be formalized as learning some suitable model $\mathcal{C} : 2^\mathcal{S} \mapsto \mathcal{L}$ of the associations between the occurrence of the chosen features in the XML trees of $\mathcal{D}$ and the class labels of the same XML trees. The resulting classifier $\mathcal{C}$ is useful to predict the unknown class of a previously unseen XML tree $\mathbf{t}'$, on the basis of the features occurring in $\mathbf{t}'$.

Ideally, the efficiency and scalability of XML classification should suffer neither from the dimensionality of $\mathcal{S}$ (i.e., the number $|\mathcal{S}|$ of features), nor from the costs of the operations for the manipulation of tree-like structures. To meet such requirements, the dimensionality of the feature space corresponding to $\mathcal{S}$ is sensibly reduced in subsection 4.2. Additionally, XML data is represented in a convenient transactional form, that avoids the manipulation of XML trees.

The idea behind the transactional representation is that by looking at the elements in $\mathcal{S}$ as binary features, the available XML data can be projected into a feature space, wherein the occurrence of the individual features within each XML tree is explicitly represented. More precisely, if $\mathcal{S}$ denotes the se-

lected collection of features, the XML trees from $\mathcal{D}$ can be modeled as transactions over a feature space $\mathcal{F} \triangleq \{\mathcal{F}_{p.w} | p.w \in \mathcal{S}\}$. Here, the generic feature $\mathcal{F}_{p.w}$ is a boolean attribute, whose value indicates the presence/absence of the corresponding feature $p.w$ of $\mathcal{S}$ within the individual XML trees.

Let $\mathbf{x}^{(\mathbf{t})}$ be the transactional representation of an XML tree $\mathbf{t}$. The value of each attribute $\mathcal{F}_{p.w}$ within $\mathbf{x}^{(\mathbf{t})}$ is true if $p.w$ is a feature of $\mathbf{t}$, otherwise it is false. Hence, $\mathbf{x}^{(\mathbf{t})}$ can be modeled as a proper subset of $\mathcal{F}$, namely $\mathbf{x}^{(\mathbf{t})} \triangleq \{\mathcal{F}_{p.w} \in \mathcal{F} \,|\, p.w \preceq \mathbf{t}\}$, with the meaning that the features explicitly present in $\mathbf{x}^{(\mathbf{t})}$ take value true, whereas the others assume value false. The original database $\mathcal{D}$ can hence be represented in transactional form as $\mathcal{D}^{(F)} = \{\mathbf{x}^{(\mathbf{t}_1)}, \ldots, \mathbf{x}^{(\mathbf{t}_N)}\}$, whereas the class associated with the generic transaction is denoted as $class(\mathbf{x}^{(\mathbf{t})})$. Hereafter, to keep notation uncluttered, the transactional database and the generic transaction will be denoted, respectively, as $\mathcal{D}$ and $\mathbf{x}$.

In this paper, XML classification is approached through associative classification (Liu et al., 1998), a powerful enhancement of conventional rule learning, that results from the integration of two fundamental tasks in data mining, namely, association rule mining and classification. Associative classifiers retain the advantages of traditional rule-based classification models (i.e., interpretability, expressiveness and high effectiveness) and, also, tend to achieve a better predictive performance (Xin and Han, 2003).

The necessary concepts concerning associative classification in the domain of the transactional representation of the XML trees are formalized next.

The notion of class association rule is the starting point.

Definition 2.3. ***Class Association Rule.*** *Let $\mathcal{F}$ be a feature space, deriving from the selection of certain features of the XML data. Also, assume that $\mathcal{D}$ (the so-called training data) is a database of XML trees represented as transactions over $\mathcal{F}$ and that $\mathcal{L}$ is a set of class labels. A class association rule (or, equivalently, a CAR) $\mathbf{r} : \mathbf{I} \rightarrow c$ is a pattern that catches the association (i.e. the co-occurrence) in $\mathcal{D}$ of some subset I of $\mathcal{F}$ with a class label c belonging to $\mathcal{L}$. I and c are said to be, respectively, the antecedent and consequent of the CAR.* □

Essentially, a CAR relates the occurrence of a certain combination of features in a transaction corresponding to an XML tree with one particular class.

A rule $\mathbf{r} : \mathbf{I} \rightarrow c$ is said to *cover* a (labeled or unlabeled) transaction $\mathbf{x} \in \mathcal{D}$ (and, dually, $\mathbf{x}$ is said to trigger or fire $\mathbf{r}$) if the condition $I \subseteq \mathbf{x}$ holds. The set $\mathcal{D}_{\mathbf{r}}$ of transactions covered by $\mathbf{r}$ is defined as $\mathcal{D}_{\mathbf{r}} = \{\mathbf{x} \in \mathcal{D} \,|\, I \subseteq \mathbf{x}\}$.

The notions of support and confidence are employed to define the interestingness of a rule $\mathbf{r}$.

Definition 2.4. ***CAR Support and Confidence.*** *A transaction $\mathbf{x} \in \mathcal{D}$* supports *a CAR $\mathbf{r} : \mathbf{I} \rightarrow c$ if it holds that $I \subseteq \mathbf{x}$ and $c = class(\mathbf{x})$. The support of $\mathbf{r}$, denoted as $supp(\mathbf{r})$, is the fraction of transactions in $\mathcal{D}$ that support $\mathbf{r}$. The confidence or predictive strength of $\mathbf{r}$, denoted by $conf(\mathbf{r})$, is defined as $conf(\mathbf{r}) = \frac{supp(\mathbf{r})}{supp(I)}$, where $supp(I)$ is the fraction of transactions in $\mathcal{D}$ including the subset I.* □

Hereafter, a CAR $\mathbf{r}$ is actually interesting if it meets certain minimum requirements on its support and confidence and if its antecedent and consequent are positively correlated (Arunasalam and Chawla, 2006). This avoids misleading CARs (i.e., CARs with negative correlation despite a high confidence) with skewed classes, which are often encountered in the XML domain.

Definition 2.5. ***Associative Classifier.*** *An associative classifier $\mathcal{C}$ is a disjunction $\mathcal{C} = \{\mathbf{r}_1 \vee \ldots \vee \mathbf{r}_k\}$ of interesting CARs learnt from a database $\mathcal{D}$ of labeled transactions (representing XML trees with known class labels).* □

An associative classifier is a set of CARs that assign an unlabeled XML tree (in transactional form) to a class if certain features occur in the tree. An approach to induce associative classifiers for effective XML classification is proposed next.

3 THE XCCS APPROACH

XCCS is a general framework for the associative classification of XML data, that relies on CARs to model the associations between subsets of co-occurring features and the discriminated classes.

XCCS exploits a selection of features of the available XML data for the discrimination of the individual classes. XML classification in XCCS divides into model learning and prediction. The former learns an associative classifier $\mathcal{C}$ from a database of labeled XML trees in transactional form. The latter exploits $\mathcal{C}$ to predict the class of unlabeled XML trees.

3.1 Model Learning

The model learning process in XCCS, sketched in fig. 1, receives four input parameters: a database $\mathcal{D}$ of XML trees, a set $\mathcal{S}$ of discriminatory features, a set $\mathcal{L}$ of class labels in $\mathcal{D}$ and one global threshold τ, from which the minimum support thresholds for the individual classes in $\mathcal{L}$ are derived.

MODEL-LEARNING($\mathcal{D}$,$\mathcal{L}$,$\mathcal{S}$,τ)
Input: a training dataset $\mathcal{D}$;
a set $\mathcal{S}$ of substructures of the XML trees in $\mathcal{D}$;
a set $\mathcal{L}$ of class labels in $\mathcal{D}$;
and a support threshold τ;
Output: An associative classifier $\mathcal{C} = \{r_1 \vee \ldots \vee r_k\}$;
1: let $\mathcal{F} \leftarrow \{\mathcal{F}_\mathbf{s} | \mathbf{s} \in \mathcal{S}\}$ be the feature space;
2: $\mathbf{R} \leftarrow \emptyset$;
3: $\mathcal{D}' \leftarrow \emptyset$;
4: **for each** $\mathbf{t} \in \mathcal{D}$ **do**
5: $\mathbf{x} \leftarrow \{\mathcal{F}_\mathbf{s} | \mathcal{F}_\mathbf{s} \in \mathcal{F}, \mathbf{s} \preceq \mathbf{t}\}$;
6: $\mathcal{D}' \leftarrow \mathcal{D}' \cup \{\mathbf{x}\}$;
7: **end for**
8: $\mathbf{R} \leftarrow$ MINECARS($\mathcal{F}$, $\mathcal{D}'$, τ);
9: $\mathcal{C} \leftarrow$ PRUNE($\mathbf{R}$);
10: RETURN $\mathcal{C}$

Figure 1: Model learning in XCCS.

Model learning preliminarily involves the definition (at line 1) of the space $\mathcal{F}$ of features related to the elements of $\mathcal{S}$ as well as the mapping of the individual XML trees in $\mathcal{D}$ to as many corresponding transactions over $\mathcal{F}$ (lines 4-7). The MINECARS procedure is then used to discover a potentially large set $\mathbf{R}$ of CARs targeting the classes in $\mathcal{L}$. The rule set $\mathbf{R}$ is eventually distilled into a compact associative classifier $\mathcal{C}$ through the pruning method PRUNE.

A detailed treatment of the MINECARS and PRUNE steps is provided, respectively, in subsections 3.1.1 and 3.1.2.

3.1.1 Mining the Class Association Rules

MINECARS is an Apriori-based procedure, that searches for meaningful CARs in the training data $\mathcal{D}$. MINECARS enhances the basic Apriori algorithm (Agrawal and Srikant, 1994) by incorporating two effective mechanisms, i.e., multiple minimum class support (Liu et al., 2000) and complement class support (Arunasalam and Chawla, 2006), with which to distill, within each class in $\mathcal{D}$, an appropriate number of CARs with a positive correlation between their antecedents and consequents. The exploitation of these mechanism is particularly useful when the distribution of classes in $\mathcal{D}$ is skewed. In the absence of suitable expedients, class imbalance typically limits the extraction of a suitable number of CARs from the less frequently occurring classes and negatively acts on the correlation of CAR antecedents and consequents, up to the point of identifying misleading CARs (i.e. with a negative correlation).

Figure 2 shows the scheme of MINECARS, which divides into frequent itemset discovery (lines P1-P18) and CAR generation (lines P19- P26). In the ongoing discussion, an itemset is a subset of structural features from the space $\mathcal{F}$.

Frequent itemset discovery starts (at line P3) with $\mathbf{C}_1$, a set of candidate 1-itemsets, consisting of one structural feature from $\mathcal{F}$ and a class label from $\mathcal{L}$. At the generic iteration, MINECARS builds a set $\mathbf{L}_k$ of frequent k-itemsets from $\mathbf{L}_{k-1}$. Two steps are performed for this purpose. The *join step* (at line P14) involves joining $\mathbf{L}_{k-1}$ with itself to yield $\mathbf{C}'_k$, a collection of candidate k-itemsets. Notice that this requires joining pairs of frequent $k-1$-itemsets with identical class labels. The well-known *Apriori* property, according to which an unfrequent itemset cannot have frequent supersets, is then used (at line P15) to drop from $\mathbf{C}'_k$ those k-itemsets with at least one $k-1$-subset that is not in $\mathbf{L}_{k-1}$. The *support counting step* (lines P5- P12) involves counting the occurrences of the surveyed candidate itemsets in $\mathbf{C}_k$ by scanning the training data $\mathcal{D}$. Those candidates whose support exceeds a class-specific threshold are considered to be frequent and retained within $\mathbf{L}_k$.

MINECARS halts when no more frequent itemsets can be discovered.

Multiple minimum class support (Liu et al., 2000) is employed at line P13 to automatically adjust the global minimum support threshold τ, supplied by the user, to the minimum support threshold specific for each class. Essentially, the generic candidate itemset c is frequent if its support is over $\tau \cdot supp(class(c))$, the adjusted minimum support threshold for $class(c)$.

If class distribution is skewed, multiple minimum class support implements a first stage of focused pruning, that dynamically assigns a higher minimum support threshold to more frequent classes (which prevents from yielding several overfitting itemsets) and a lower minimum support threshold to less frequent classes (which enforces the generation of an appropriate number of itemsets).

Instead, complement class support (Arunasalam and Chawla, 2006) is used in the CAR generation stage, to avoid the specification of a global minimum confidence threshold. In particular, a specific property of complement class support (shown in (Arunasalam and Chawla, 2006)) is exploited at line P22 to automatically identify a class-specific minimum confidence threshold. According to such a property, a rule $\mathbf{r} : \mathbf{I} \rightarrow c$ is such that I and c are positively correlated if and only if $conf(I \rightarrow c) > \frac{\sigma(c)}{|\mathcal{D}|}$, where $\sigma(c)$ is the overall number of occurrences of class c in $\mathcal{D}$. Therefore, the CARs whose confidence exceeds (at line P22) the minimum threshold corresponding to their targeted class are guaranteed to be positively correlated. Thus, both confidence and positive correlation between rule components can be verified without additional parameters or further correlation analysis.

When classes are skewed, the dynamic setting of a

MINECARS($\mathcal{F}$,$\mathcal{D}$,τ)
Input: a finite set of boolean attributes $\mathcal{F}$;
a training set $\mathcal{D}$ of transactions representing XML trees;
and a support threshold τ;
Output: a set $\mathbf{R}$ of class association rules;
/* Frequent itemset discovery */
P1: $I \leftarrow \emptyset, k \leftarrow 1$;
P2: Let $\mathcal{L}$ be the set of class labels in $\mathcal{D}$;
P3: Let $\mathbf{C}_1 \leftarrow \{c | \exists \mathcal{F}_s \in \mathcal{F}, l \in \mathcal{L}$ such that $c = \{\mathcal{F}_s\}, class(c) = l\}$;
P4: **while** $\mathbf{C}_k \neq \emptyset$ **do**
P5: **for** each candidate candidate itemset $c \in \mathbf{C}_k$ **do**
P6: $supp(c) \leftarrow 0$;
P7: **end for**
P8: **for** $\mathbf{x} \in \mathcal{D}$ **do**
P9: **for** $c \in \mathbf{C}_k$ such that $class(c) = class(\mathbf{x})$ and $c \subseteq \mathbf{x}$ **do**
P10: $supp(c) \leftarrow supp(c) + \frac{1}{|\mathcal{D}|}$;
P11: **end for**
P12: **end for**
P13: $\mathbf{L}_k \leftarrow \{c \in \mathbf{C}_k | supp(c) > \tau \cdot supp(class(c))\}$;
P14: $\mathbf{C}'_{k+1} \leftarrow \{c_i \cup c_j | c_i, c_j \in \mathbf{L}_k \wedge class(c_i) = class(c_j) \wedge |c_i \cup c_j| = k+1\}$;
P15: $\mathbf{C}_{k+1} \leftarrow \{c \in \mathbf{C}'_{k+1} | \forall c' \subset c$ with $|c'| = k$, it holds that $c' \in \mathbf{L}_k\}$;
P16: $k \leftarrow k+1$;
P17: **end while**
P18: $I \leftarrow \cup_k \mathbf{L}_k$;
/* CAR generation */
P19: $\mathbf{R} \leftarrow \emptyset$;
P20: **for** each frequent itemset $I \in I$ **do**
P21: create rule $\mathbf{r} : I \rightarrow class(I)$;
P22: **if** $conf(\mathbf{r}) > \frac{\sigma(class(I))}{|\mathcal{D}|}$ **then**
P23: $\mathbf{R} \leftarrow \mathbf{R} \cup \{\mathbf{r}\}$;
P24: **end if**
P25: **end for**
P26: RETURN $\mathbf{R}$;

Figure 2: The MINECARS procedure.

class-specific minimum confidence threshold acts as a second stage of focused pruning, that ensures the discovery of discriminative rules targeting the unfrequent classes and still avoids an overwhelming number of rules from the predominant classes.

3.1.2 Learning an Associative-classification Model

Due to the inherently combinatorial nature of the associative patterns, MINECARS may yield a large number of CARs, which are likely to overfit the training data and provide contrasting predictions. To avoid such issues, a compact and accurate classifier is distilled from the rule set $\mathbf{R}$ through the covering method PRUNE, illustrated in fig. 3.

PRUNE initially orders (at line M1) the available CARs according to the total order $\ll$, which is inspired to the one introduced in (Liu et al., 1998). Precisely, given any two CARs $\mathbf{r}_i, \mathbf{r}_j \in \mathbf{R}$, $\mathbf{r}_i$ precedes $\mathbf{r}_j$, which is denoted by $\mathbf{r}_i \ll \mathbf{r}_j$, if (*i*) $conf(\mathbf{r}_i)$ is greater than $conf(\mathbf{r}_j)$, or (*ii*) $conf(\mathbf{r}_i)$ is the same as $conf(r_j)$, but $supp(\mathbf{r}_i)$ is greater than $supp(\mathbf{r}_j)$, or (*iii*) $conf(\mathbf{r}_i)$ is the same as $conf(r_j)$ and $supp(\mathbf{r}_i)$ is identical to $supp(\mathbf{r}_j)$, but $length(\mathbf{r}_i)$ is less than $length(\mathbf{r}_j)$. The length of a CAR $\mathbf{r} : \mathbf{I} \rightarrow c$ is the number of features in the antecedent of $\mathbf{r}$, i.e., $length(\mathbf{r}) = |I|$.

If two CARs $\mathbf{r}_i, \mathbf{r}_j$ have equal confidence, support and length, then $\mathbf{r}_i \ll \mathbf{r}_j$ if $\mathbf{r}_i$ was generated before $\mathbf{r}_j$.

A covering process (lines M4- M19) then seeks a compact classifier $\mathcal{C}$, consisting of a minimal number of CARs from $\mathbf{R}$, that attain a high predictive accuracy over unlabeled transactions (representing unclassified XML trees).

The covering process attempts the maximization of the effectiveness $F(\mathcal{C})$ of the resulting classifier $\mathcal{C}$ across all classes. $F(\mathcal{C})$ is evaluated in terms of the macro-averaged F-measure (Manning et al., 2008) of $\mathcal{C}$, which is defined as follows

$$F(\mathcal{C}) = \frac{1}{|\mathcal{L}|} \sum_{c \in \mathcal{L}} F^{(c)}(\mathcal{C})$$

where $F^{(c)}(\mathcal{C})$ is the effectiveness (or, also, the predictive performance) of $\mathcal{C}$ over the generic class c, described below. $F(\mathcal{C})$ assigns a same relevance to the effectiveness of $\mathcal{C}$ over the different classes, regardless of the occurrence frequency of the individual classes in the training data. This is especially useful in the presence of class imbalance, since $F^{(c)}(\mathcal{C})$ is not dominated by the predictive performances of $\mathcal{C}$ over the most frequent classes across the transactions.

The covering process increases $F(\mathcal{C})$ by separately acting on each $F^{(c)}(\mathcal{C})$, through the selection (at line M6) of CARs from $\mathbf{R}$ that, when appended to $\mathcal{C}$ (at line M10), improve the predictive performance of the resulting classifier over c without a significant loss in compactness.

For each class c, covering scans (according the $\ll$ order) the different CARs of $\mathbf{R}$ that target c. A CAR $\mathbf{r} : \mathbf{I} \rightarrow c$ is appended to $\mathcal{C}$ if $F^{(c)}(\mathcal{C} \cup \{\mathbf{r}\})$ is greater than $F^{(c)}(\mathcal{C})$ (at line M7). In this case, $\mathbf{r}$ is removed from $\mathbf{R}$ (at line M9) and all transactions covered by $\mathbf{r}$ are dropped from $\mathcal{D}$ (at line M12).

Notice that $F^{(c)}(\mathcal{C})$ is 0 while $\mathcal{C}$ does not include any CARs targeting c.

The total order established over $\mathbf{R}$ (at line M1) plays a key role while covering the generic class c: it ensures that the CARs predicting c with highest implicative strength are appended to $\mathcal{C}$ since the early stages of the covering process, which is beneficial for $F^{(c)}$. Moreover, as covering of class c proceeds, the consequent removal of transactions from $\mathcal{D}$ operates as a pruning method, which increasingly tends to prevent from retaining in $\mathcal{C}$ those CARs targeting c, that have not yet been considered. This positively acts on

both the effectiveness and the compactness of $\mathcal{C}$ since, according to the $\ll$ order, such CARs predict c with either a lower implicative strength or a higher specificity.

In particular, pruning specific CARs is useful to filter redundancy (Li et al., 2001) away from $\mathcal{C}$. Therein, let $\mathbf{r} : \mathbf{I} \rightarrow c$ and $\mathbf{r}' : I' \rightarrow c$ be two redundant CARs of $\mathbf{R}$, such that $I \subset I'$ (i.e. $\mathbf{r}'$ is more specific than $\mathbf{r}$) and the implicative strength of $\mathbf{r}'$ is lower than that of $\mathbf{r}$. According to the order of the CARs, it holds that $\mathbf{r} \ll \mathbf{r}'$ (i.e., $\mathbf{r}$ precedes $\mathbf{r}'$ in $\mathbf{R}$). Therefore, if $\mathbf{r}$ is added to $\mathcal{C}$, the consequent removal from $\mathcal{D}$ of $\mathcal{D}_{\mathbf{r}}$ (i.e. the set of transactions covered by $\mathbf{r}$) also involves the elimination of $\mathcal{D}_{\mathbf{r}'}$, since $\mathcal{D}_{\mathbf{r}'}$ is actually a subset of $\mathcal{D}_{\mathbf{r}}$. As a consequence, when covering subsequently considers $\mathbf{r}'$, the latter will be unable to improve $\mathrm{F}^{(c)}$ and, thus, will be discarded. Redundancy avoidance strongly contributes to the compactness of $\mathcal{C}$ (especially when the size of $\mathbf{R}$ is large), in cooperation with the information-theoretic scheme at the end of this section. In the envisaged cooperation, pruning specific CARs avoids redundancy, whereas the latter scheme is used to evaluate whether the gain in predictive performance, due to the addition of *non-redundant* CARs to $\mathcal{C}$, is worth the consequent loss in compactness.

Notice that the different classes are separately covered (at line M4) in increasing order of their occurrence frequency, to avoid that transactions belonging to less frequent classes are removed from $\mathcal{D}$ while covering other classes with higher occurrence frequency. This would have the undesirable effect of avoiding an appropriate evaluation of the gain in effectiveness due to the addition to $\mathcal{C}$ of CARs targeting the foresaid less frequent classes.

Covering halts when either there are no more CARs to consider (which is caught, for each class c, at line M5), or all training transactions have been covered (which is caught at line M15), or the predictive performance of $\mathcal{C}$ cannot be further increased (which is caught, for each class c, at line M5).

The generic $\mathrm{F}^{(c)}$ summarizes two further measures of class-specific effectiveness, i.e., the degree of precision $\mathrm{P}^{(c)}$ and recall $\mathrm{R}^{(c)}$ of classifier $\mathcal{C}$ in class c:

$$\mathrm{F}^{(c)}(\mathcal{C}) = 2\left[\frac{1}{\mathrm{P}^{(c)}(\mathcal{C})} + \frac{1}{\mathrm{R}^{(c)}(\mathcal{C})}\right]^{-1}$$

An increase in $\mathrm{F}^{(c)}$, due to the addition of a CAR $\mathbf{r}$ to the current classifier $\mathcal{C}$, means that $\mathbf{r}$ produces an acceptable improvement of the predictive performance of $\mathcal{C}$, ascribable to a sensible gain in at least one between $\mathrm{P}^{(c)}$ and $\mathrm{R}^{(c)}$.

Precision $\mathrm{P}^{(c)}(\mathcal{C})$ is the exactness of $\mathcal{C}$ within class c, i.e., the proportion of transactions that are actually of class c among the ones assigned by $\mathcal{C}$ to class c. Recall $\mathrm{R}^{(c)}(\mathcal{C})$ is instead the completeness of $\mathcal{C}$ within c, i.e., the fraction of transactions of class c that are correctly predicted by $\mathcal{C}$. Formally, let $\mathcal{D}_{\mathcal{C}}^{(c)} = \{\mathbf{x} \in \mathcal{D} \,|\, \exists \mathbf{r} \in \mathcal{C}, \mathbf{r} : \mathbf{I} \rightarrow c, I \subseteq \mathbf{x}\}$ be the set of transactions covered by the CARs of $\mathcal{C}$ predicting class c and $p_{\mathcal{C}}^{(c)} = \{\mathbf{x} \in \mathcal{D} \,|\, \exists \mathbf{r} \in \mathcal{C}, \mathbf{r} : \mathbf{I} \rightarrow c, I \subseteq \mathbf{x}, class(\mathbf{x}) = c\}$ be the set of transactions correctly assigned by $\mathcal{C}$ to class c. Also, assume that $\sigma(c)$ is the overall number of transactions of class c. Precision $\mathrm{P}^{(c)}(\mathcal{C})$ and recall $\mathrm{R}^{(c)}(\mathcal{C})$ are defined as reported below

$$\mathrm{P}^{(c)}(\mathcal{C}) = \frac{|p_{\mathcal{C}}^{(c)}|}{|\mathcal{D}_{\mathcal{C}}^{(c)}|} \qquad \mathrm{R}^{(c)}(\mathcal{C}) = \frac{|p_{\mathcal{C}}^{(c)}|}{\sigma(c)}$$

Precision $\mathrm{P}^{(c)}(\mathcal{C})$ and recall $\mathrm{R}^{(c)}(\mathcal{C})$ provide complementary information on the effectiveness of $\mathcal{C}$ over c. Indeed, an improvement in precision alone, achieved by appending $\mathbf{r}$ to $\mathcal{C}$, would not say anything on the corresponding variation in the recall of $\mathcal{C} \cup \{\mathbf{r}\}$. Dually, an improvement in recall alone would not say anything on the corresponding variation in the precision of $\mathcal{C} \cup \{\mathbf{r}\}$.

The simultaneous increase of both precision and recall is a challenging issue in the design of algorithms for learning classification models, since it often happens that a gain in recall corresponds to a loss in precision and vice-versa. $\mathrm{F}^{(c)}(\mathcal{C})$ is the harmonic mean of $\mathrm{P}^{(c)}(\mathcal{C})$ and $\mathrm{R}^{(c)}(\mathcal{C})$ and, hence, it is always closer to the smaller between the two. Therefore, an improvement of $\mathrm{F}^{(c)}(\mathcal{C} \cup \{\mathbf{r}\})$ with respect to $\mathrm{F}^{(c)}(\mathcal{C})$ ensures that an increase in recall due to the addition of $\mathbf{r}$ to $\mathcal{C}$ is not vanished by a serious loss in precision.

Let $\mathcal{D}_{\mathbf{r}}$ be set of transactions left in $\mathcal{D}$, that are covered by $\mathbf{r} : \mathbf{I} \rightarrow c$ (at line M11). Also, assume that $p_{\mathbf{r}}^{(c)}$ is the subset of those transactions in $\mathcal{D}_{\mathbf{r}}$ correctly classified by $\mathbf{r}$ into class c, i.e., $p_{\mathbf{r}}^{(c)} = \{\mathbf{x} \in \mathcal{D}_{\mathbf{r}} | class(\mathbf{x}) = c\}$. The updated values of precision $\mathrm{P}^{(c)}(\mathcal{C} \cup \{\mathbf{r}\})$ and recall $\mathrm{R}^{(c)}(\mathcal{C} \cup \{\mathbf{r}\})$, resulting from the addition of $\mathbf{r}$ to $\mathcal{C}$, are incrementally computed from $\mathrm{P}^{(c)}(\mathcal{C})$ and $\mathrm{R}^{(c)}(\mathcal{C})$ as follows

$$\mathrm{P}^{(c)}(\mathcal{C} \cup \{\mathbf{r}\}) = \frac{|p_{\mathcal{C}}^{(c)}| + |p_{\mathbf{r}}^{(c)}|}{|D_{\mathcal{C}}^{(c)}| + |\mathcal{D}_{\mathbf{r}}|}$$

$$\mathrm{R}^{(c)}(\mathcal{C} \cup \{\mathbf{r}\}) = \mathrm{R}^{(c)}(\mathcal{C}) + \frac{|p_{\mathbf{r}}^{(c)}|}{\sigma(c)}$$

When covering ends (line M19), the resulting classifier $\mathcal{C}$ is a list of predictive CARs grouped by the targeted class. The individual groups of CARs appear in $\mathcal{C}$ in increasing order of the occurrence frequency of the targeted class. Moreover, within each

```
PRUNE(R,D,L)
  Input: a set R of CARs;
         a set D of transactions;
         a set L of class labels in D;
  Output: a classifier C;
     /* Rule ordering according to the devised total order ≪ */
M1:  R ← ORDER(R);
M2:  C ← ∅;
M3:  T ← D;
M4:  for each c ∈ L in increasing order of occurrence frequency do
M5:    while there are still CARs in R that target c do
M6:      choose the next rule r : I → c from R;
M7:      if F^(c)(C ∪ {r}) > F^(c)(C) then
M8:        cur_length ← length(C ∪ {r});
M9:        R ← R − {r};
M10:         C ← C ∪ {r};
M11:         D_r ← {x ∈ D | I ⊆ x};
M12:         D ← D − D_r;
M13:         min_length ← cur_length;
M14:     end if
M15:     if |D| = 0 then
M16:       continue at line M20;
M17:     end if
M18:   end while
M19: end for
M20: if |D| > 0 then
M21:   c* ← argmax_{c∈L} supp(c, D);
M22: else
M23:   c* ← argmax_{c∈L} supp(c);
M24: end if
M25: C ← C ∪ {∅ → c*};
M26: RETURN C;
```

Figure 3: The PRUNE procedure.

group, the CARs reflect the total order $\ll$ established over $\mathbf{R}$.

As it is said in (Ning et al., 2006), the class-based ordering of the CARs in $\mathcal{C}$ confers to the classifier a high interpretability, that would not be obtained if the same CARs were sorted in $\mathcal{C}$ according to the $\ll$ relationship. Indeed, in this latter case, the meaning of each CAR $\mathbf{r}$ in $\mathcal{C}$ would involve the negation of any other CAR $\mathbf{r}'$ in $\mathcal{C}$ such that $\mathbf{r}' \ll \mathbf{r}$. This would clearly reduce the comprehension of the CARs sited at the bottom of $\mathcal{C}$, especially if the size of $\mathcal{C}$ is large. Class-based ordering has been largely adopted in the design of seminal rule-induction algorithms.

To conclude the discussion on PRUNE, the mutual exclusiveness and the exhaustive coverage of the CARs of any resulting classifier $\mathcal{C}$ must be touched.

A rule-based classifier is mutually exclusive if each input triggers no more than one rule. Generally, such a property does not hold for $\mathcal{C}$. Indeed, it is actually possible that multiple CARs are triggered by a same transaction. This is clearly undesirable because (some of) the triggered CARs may provide contrasting predictions. This problem is overcome in sec. 3.2.

Instead, the addition to $\mathcal{C}$ (at line M25) of a default rule $\emptyset \rightarrow c^*$ ensures exhaustive coverage, i.e., that every transaction is covered by at least one CAR of $\mathcal{C} \cup \{\emptyset \rightarrow c^*\}$. In particular, the default rule covers all those transactions uncovered by the CARs of $\mathcal{C}$ and assigns them to a suitable class c^*. This guarantees a maximum recall with very poor precision in class c^*. To attenuate the loss in precision, c^* can be reasonably chosen (at line M23) to be the class with highest occurrence frequency in the training data, which ensures the highest precision for the default rule. Depending on the overall coverage of $\mathcal{C}$, there are two alternative possibilities for the choice of the default class c^*. If there are still uncovered transactions after the termination of the covering process (at line M20), c^* is selected (at line M21) as the class with maximum occurrence frequency $supp(c, \mathcal{D})$ in the residual training data $\mathcal{D}$. Otherwise, if all transactions have been covered, c^* is chosen (at line M23) to be the class with highest occurrence frequency in the whole training data. In both cases, ties can be arbitrarily broken.

3.2 Prediction

Let $\mathcal{C}$ be an associative classifier induced by XCCS at the end of model-learning phase of fig. 1. Also, assume that $\mathbf{t}$ is an unlabeled (i.e. an unclassified) XML tree, whose transactional representation over the underlying feature space $\mathcal{F}$ is $\mathbf{x}$. The class predicted by $\mathcal{C}$ for $\mathbf{t}$ is $\mathcal{C}(\mathbf{x})$. To avoid conflicting predictions from multiple triggered CARs, $\mathcal{C}(\mathbf{x})$ is provided by the first CAR of $\mathcal{C}$ that covers $\mathbf{x}$.

4 EVALUATION

The empirical behavior of XCCS is studied in order to comparatively evaluate the effectiveness of XCCS across different domains;

All tests are performed on a Linux machine, with an Intel Core 2 Duo cpu, 4Gb of memory and 2Ghz of clock speed.

4.1 Data Sets

The behavior of XCCS is tested over several real XML data sets. Synthetic XML corpora are not considered for experimental purposes, since these are generally unlikely to provide coherent textual information in natural language.

Macro-averaged effectiveness results are obtained by performing a stratified 10-fold cross validation on the transactional representation of each data set.
We choose four real-world XML data sets, that

include textual information and are characterized by skewed distributions of the classes of XML documents.

Wikipedia is an XML corpus proposed in the INEX contest 2007 (Denoyer and Gallinari, 2008) as a major benchmark for XML classification and clustering. The corpus groups 96,000 XML documents representing very long articles from the digital encyclopedia. The XML documents are organized into 21 classes (or thematic categories), each corresponding to a distinct *Wikipedia Portal*. A challenging aspect of the corpus is the ambiguity of certain pairs of classes such as, e.g., `Portal:Pornography` and `Portal:Sexuality` or `Portal:Chistianity` and `Portal:Spirituality` (Denoyer and Gallinari, 2008).

IEEE is a reference text-rich corpus, presented in (Denoyer and Gallinari, 2007), that includes 12,107 XML documents representing full articles. These are organized into 18 classes corresponding to as many IEEE Computer Society publications: 6 `Transactions` and 12 other journals. A same thematic can be treated into two distinct journals.

DBLP is a bibliographic archive of scientific publications on computer science (`http://dblp.unitrier.de/xml/`). The archive is available as one very large XML file with a diversified structure. The whole file is decomposed into 479,426 XML documents corresponding to as many scientific publications. These individually belong to one of 8 classes: `article` (173,630 documents), `proceedings` (4,764 documents), `mastersThesis` (5 documents), `incollection` (1,379 documents), `inproceedings` (298,413 documents), `book` (1,125 documents), `www` (38 documents), `phdthesis` (72 documents). The individual classes exhibit differentiated structures, despite some overlap among certain document tags (such as *title*, *author*, *year* and *pages*), that occur in (nearly) all of the XML documents.

The *Sigmod* collection groups 988 XML documents (i.e., articles from Sigmod Record) complying to three different class DTDs: `IndexTermsPage`, `OrdinaryIssue` and `Proceedings`. These classes contain, respectively, 920, 51 and 17 XML documents. Such classes have diversified structures, despite the occurrence of some overlapping tags, such as *volume*, *number*, *authors*, *title* and *year*.

4.2 Preprocessing

The high dimensionality (i.e. cardinality) of the feature space $\mathcal{S}$ may be a concern for the time efficiency and the scalability of model induction. In particular, when the classes of XML trees cannot be discriminated through the structural information alone and, hence, the content information must necessarily be taken into account, the number of XML features likely becomes very large if the XML documents contain huge amounts of textual data.

To reduce the dimensionality of the feature space $\mathcal{S}$, the available XML data is preprocessed into two steps.

The first step addresses the textual information of the XML data and sensibly reduces the overall number of distinct terms in the leaves of the XML trees through token extraction, stop-word removal and stemming.

Dimensionality reduction is performed at the second step both to reduce overfitting and to ensure a satisfactory behavior of XCCS in terms of efficiency, scalability and compactness of the induced classifiers. The idea is partitioning $\mathcal{S}$ into groups of XML features, that discriminate the individual classes in a similar fashion. For this purpose, we explicitly represent the discriminatory behavior of the features in $\mathcal{S}$ and then group the actually discriminatory features through distributional clustering (Baker and McCallum, 1998). In particular, the discriminatory behavior of each feature $p.w$ in $\mathcal{S}$ is represented as an array $\mathbf{v}_{p.w}$ with as many entries as the number of classes in $\mathcal{L}$. The generic entry of $\mathbf{v}_{p.w}$ is the probability $P(c|p.w)$ of class c given the XML feature $p.w$. Clearly, $P(c|p.w) = \frac{P(p.w|c)P(c)}{P(p.w)}$ by Bayes rule, where probabilities $P(p.w|c)$, $P(c)$ and $P(p.w)$ are estimated from the data. Before clustering features, noisy (i.e. non discriminatory) features are removed from $\mathcal{S}$. Specifically, a feature $p.w$ is noisy if there is no class c in $\mathcal{L}$, that is positively correlated with $p.w$ according to chi-square testing at a significance level 0.05. The arrays relative to the remaining features of $\mathcal{S}$ are then grouped through distributional clustering into a desired number of feature clusters with similar discriminatory behavior.

Eventually, an aggressive compression of the original feature space $\mathcal{S}$ is achieved by replacing each feature $p.w$ of $\mathcal{S}$ with a respective synthetic feature, i.e., the label of the cluster to which $\mathbf{v}_{p.w}$ belongs. Distributional clustering efficiently compresses the feature space by various orders of magnitude, while still enabling a significantly better classification performance than several other established techniques for dimensionality reduction (Baker and McCallum, 1998).

4.3 Classification Effectiveness

We compare XCCS against several other established competitors in terms of classification effectiveness.

Both direct and indirect comparisons are performed.

The direct comparisons involve three competing classification approaches, that produce rule-based classifiers, i.e., XRULES (Zaki and Aggarwal, 2006), CBA (Liu et al., 1998) as well as CPAR (Xin and Han, 2003). These competitors are publicly available and, thus, can be compared against XCCS on each XML data set.

XRULES is a state-of-the-art competitor, that admits multiple cost-models to evaluate classification effectiveness. For each data set, we repeatedly train XRULES to suitably tune the minimum support threshold for the frequent subtrees to find in the various classes and, then, report the results of the cost model allowing the best classification performance.

CBA and CPAR are two seminal techniques for learning associative classifiers. CBA and CPAR are included among the competitors of XCCS to compare the effectiveness of the three distinct approaches to associative classification at discriminating classes in (high-dimensional) transactional domains.

To evaluate CBA and CPAR, we use the implementations from (Coenen, 2004). In all tests, both CBA and CPAR are trained on the transactional representations of the XML data sets used to feed XCCS. Again, CBA and CPAR are repeatedly trained on the transactional representation of each XML data set, in order to suitably tune their input parameters. For every data set, we report the results of the most effective classifiers produced by CBA and CPAR.

Through preliminary tests we noticed that, in all tests, a satisfactory behavior of XCCS can be obtained by fixing the support threshold τ of fig. 1 to 0.1. This is essentially due to the adoption of the minimum class support (Liu et al., 2000) (discussed in section 3.1.1) in the MINECARS procedure of fig. 2.

Fig. 4 summarizes the effectiveness of the chosen competitors across the selected data sets.

Columns **Size** and **#C** indicate, respectively, the number of XML documents and classes for each corpus. Column **Model** identifies the competitors. **Rules** is the rounding off of the average number of rules of a classifier in the stratified 10-fold cross validation.

The effectiveness of each classifier is measured in terms of average precision (**P**), average recall (**R**), average F-measure (**F**). More precisely, the values of **P**, **R** and **F** are averages of precision, recall and F-measure over the folds of the stratified 10-fold cross validation of classifiers on the individual data sets.

The maximum values of **P**, **R** and **F** on each data set are highlighted in bold.

Notice that we tabulate only the (best) results achieved by the approaches (de Campos et al., 2008; Murugeshan et al., 2008; Yang and Zhang, 2008; Yong et al., 2007; Xing et al., 2007) in the respective papers.

Some results were not originally measured and, hence, are reported as *N.A.* (short for *not available*). **Rules** has no sense for (de Campos et al., 2008; Murugeshan et al., 2008; Yang and Zhang, 2008; Yong et al., 2007; Xing et al., 2007) and, thus, its entry in the corresponding rows is left blank. The symbol – that appears in three rows of fig. 4 reveals that XRULES did not successfully complete the tests over *Wikipedia*, *IEEE* and *DBLP*. The enumeration of the frequent embedded subtrees within each class and the consequent generation of predictive structural rules (satisfying the specified level of minimum class-specific support) are very time-expensive steps of XRULES, especially when the underlying number of XML documents is (very) large. In all completed tests, XRULES is less effective than XCCS. In addition, the huge number of rules produced by XRULES makes the resulting classification models difficult to understand (and, hence, hardly actionable) in practice. . The classification performance of CBA is inferior than that of XCCS on the selected data sets. Moreover, as discussed in sec. 3.1.2, interpreting CBA classifiers may be cumbersome, since their rules are not ordered by the targeted class. CPAR delivers a satisfactory classification performance on the chosen XML corpora. Nonetheless, CPAR is still less effective and compact than XCCS. The approaches (de Campos et al., 2008; Murugeshan et al., 2008; Yang and Zhang, 2008) and (Yong et al., 2007; Xing et al., 2007) exhibit generally inferior classification performances than XCCS on the *Wikipedia* and *IEEE* corpora, respectively.

To conclude, XCCS consistently induces the most effective classifiers on the chosen corpora. As far as compactness is concerned, such classifiers are generally comparable to the ones induced by CBA and significantly more compact than the ones induced by XRULES and CPAR. The effectiveness of XCCS confirms its general capability at handling XML data with skewed class distributions.

5 CONCLUSIONS AND FURTHER RESEARCH

XCCS is a new approach to XML classification that induces clearly interpretable predictive models, which are of great practical interest for more effective and efficient XML search, retrieval and filtering. XCCS induces very compact classifiers with outperforming effectiveness from very large corpora of XML data.

Ongoing research aims to increase the discrimina-

Data	Size	#C	Model	Rules	P	R	F
Wikipedia	96,000	21	XCCS	87	**0.77**	0.78	**0.78**
			XRULES	-	-	-	-
			CBA	90	0.60	0.61	0.61
			CPAR	156	0.73	0.72	0.73
			(de Campos et al., 2008)		N.A.	N.A.	0.75
			(Murugeshan et al., 2008)		N.A.	0.76	N.A.
			(Yang and Zhang, 2008)		N.A.	**0.84**	N.A.
IEEE	12,107	18	XCCS	49	**0.73**	**0.75**	**0.74**
			XRULES	-	-	-	-
			CBA	68	0.53	0.55	0.54
			CPAR	133	0.68	0.68	0.68
			(Yong et al., 2007)		*N.A.*	*N.A.*	0.72
			(Xing et al., 2007)		*N.A.*	*N.A.*	0.60
DBLP	479426	8	XCCS	10	**1**	**1**	**1**
			XRULES	-	-	-	-
			CBA	8	0.95	0.95	0.95
			CPAR	10	0.96	0.96	0.96
Sigmod	998	3	XCCS	6	**1**	**1**	**1**
			XRULES	> 50000	0.95	0.94	0.94
			CBA	3	0.91	0.93	0.92
			CPAR	32	0.93	0.94	0.93

Figure 4: Results of the empirical evaluation.

tory power of XML features by incorporating the textual context of words in the leaves of the XML trees. Also, we are studying enhancements of model learning in XCCS, with which to induce classification rules that also consider the absence of XML features.

REFERENCES

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. In *Proc. of Int. Conf. on Very Large Data Bases*, pages 487 – 499.

Arunasalam, B. and Chawla, S. (2006). CCCS: A top-down association classifier for imbalanced class distribution. In *Proc. of Int. Conf. on Knowledge Discovery and Data Mining*, pages 517–522.

Baker, L. and McCallum, A. (1998). Distributional clustering of words for text classification. In *Proc. of ACM Int. Conf. on Research and Development in Information Retrieval*, pages 96 – 103.

Coenen, F. (2004). LUCS KDD implementations of CBA and CMAR. Dpt of Computer Science, University of Liverpool - www.csc.liv.ac.uk/ frans/KDD/Software/.

de Campos, L., Fernández-Luna, J., Huete, J., and Romero, A. (2008). Probabilistic methods for structured document classification at inex'07. In *Proc. of INitiative for the Evaluation of XML Retrieval*, pages 195 – 206.

Denoyer, L. and Gallinari, P. (2007). Report on the xml mining track at inex 2005 and inex 2006. *ACM SIGIR Forum*, 41(1):79 – 90.

Denoyer, L. and Gallinari, P. (2008). Report on the xml mining track at inex 2007. *ACM SIGIR Forum*, 42(1):22 – 28.

Garboni, C., Masseglia, F., and Trousse, B. (2006). Sequential pattern mining for structure-based xml document classification. In *Proc. of the INitiative for the Evaluation of XML Retrieval*, pages 458 – 468.

Knijf, J. D. (2007). Fat-cat: Frequent attributes tree based classification. In *Proc. of the INitiative for the Evaluation of XML Retrieval*, pages 485 – 496.

Li, W., Han, J., and Pei, J. (2001). CMAR: Accurate and efficient classification based on multiple class-association rules. In *Proc. of Int. Conf. on Data Mining*, pages 369 – 376.

Liu, B., Hsu, W., and Ma, Y. (1998). Integrating classification and association rule mining. In *Proc. of Conf. on Kwnoledge Discovery and Data Mining*, pages 80–86.

Liu, B., Ma, Y., and Wong, C. (2000). Improving an association rule based classifier. In *Proc. of Int. Conf. on Principles of Data Mining and Knowledge Discovery*, pages 504 – 509.

Manning, C., Raghavan, P., and Schütze., H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Murugeshan, M., Lakshmi, K., and Mukherjee, S. (2008). A categorization approach for wikipedia collection based on negative category information and initial descriptions. In *Proc. of the INitiative for the Evaluation of XML Retrieval*.

Ning, P., Steinbach, M., and Kumar, V. (2006). *Introduction to Data Mining*. Addison Wesley.

Theobald, M., Schenkel, R., and Weikum, G. (2003). Exploiting structure, annotation, and ontological knowledge for automatic classification of xml data. In *Proc. of WebDB Workshop*, pages 1 – 6.

Xin, X. and Han, J. (2003). CPAR: Classification based on predictive association rules. In *Proc. of SIAM Int. Conf. on Data Mining*, pages 331–335.

Xing, G., Guo, J., and Xia, Z. (2007). Classifying xml documents based on structure/content similarity. In *Proc. of the INitiative for the Evaluation of XML Retrieval*, pages 444 – 457.

Yang, J. and Zhang, F. (2008). Xml document classification using extended vsm. In *Proc. of the INitiative for the Evaluation of XML Retrieval*, pages 234 – 244.

Yi, J. and Sundaresan, N. (2000). A classifier for semi-structured documents. In *Proc. of Int. Conf. on Knowledge Discovey and Data Mining*, pages 340 – 344.

Yong, S., Hagenbuchner, M., Tsoi, A., Scarselli, F., and Gori, M. (2007). Xml document mining using graph neural network. In *Proc. of the INitiative for the Evaluation of XML Retrieval*, pages 458 – 472.

Zaki, M. and Aggarwal, C. (2006). Xrules: An effective algorithm for structural classification of xml data. *Machine Learning*, 62(1-2):137–170.